Energy-Efficient Virtual Machines Placement

Albert Philippe Marcel De La Fuente Vigliotti

DISSERTATION PRESENTED
TO THE
INSTITUTE OF MATHEMATICS AND STATISTICS
OF THE
UNIVERSITY OF SÃO PAULO
TO
FULFILL THE REQUIREMENTS OF THE DEGREE
OF
MASTER IN SCIENCE

Program: Computer Science
Supervisor: Prof. Dr. Daniel Macêdo Batista

São Paulo, April 2015

Energy-Efficient Virtual Machines Placement

Acknowledgments

First of all, I would like to thank my supervisor, Professor Daniel Macêdo Batista, who has given me the opportunity of this MsC, and provided me with his guidance. I thank the reviewers who helped me to improve this work. I would like to thank my family and friends for the support and encouragement, and finally, I want to thank God for He has never abandoned me.

Then Job replied to the LORD: ”I know that you can do anything, and no one can stop you. You asked, ’Who is this that questions my wisdom with such ignorance?’ It is I–and I was talking about things I knew nothing about, things far too wonderful for me. You said, ’Listen and I will speak! I have some questions for you, and you must answer them.’ I had only heard about you before, but now I have seen you with my own eyes. I take back everything I said, and I sit in dust and ashes to show my repentance.” (Job 42:1–6)

Resumo

DE LA FUENTE VIGLIOTTI, A. P. M.**Alocação Energeticamente Eficiente de Máquinas Virtuais**. 2015. 91 f. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2015.
Centros de processamento de dados (CPDs) são responsáveis por 1,5% do consumo mundial de energia elétrica. Esse consumo tende a ampliar diretamente o efeito estufa e a emissão de gás carbônico. Técnicas de virtualização, que já vêm sendo utilizadas em provedores de computação em nuvem, podem ser utilizadas para aumentar a eficiência energética de CPDs já que, com a virtualização, a infraestrutura nesses centros passa a permitir o compartilhamento de um mesmo hardware físico por várias máquinas virtuais (MVs). Uma alocação eficiente de MVs pode diminuir a necessidade de hardware e o consumo de energia.

Boa parte dos algoritmos de alocação de VMs existentes foca no compartilhamento de um único tipo de recurso, como o processador, ou assume que as demandas de recursos são determinísticas. Nesta dissertação de mestrado são apresentados e comparados algoritmos de alocação de VMs com o objetivo de reduzir o consumo de energia elétrica, além de serem considerados vários tipos de recursos com demandas não determinísticas. São implementados mecanismos de consolidação de VMs para reduzir o consumo de energia em CPDs e sem violar SLAs. Três algoritmos são apresentados. Os algoritmos diferem-se pela heurística implementada, sendo que dois deles baseiam-se no problema da mochila e um modela o problema utilizando computação evolutiva. Em média, em experimentos de simulação com configurações de computadores reais, os algoritmos propostos reduziram o consumo de energia a partir de 52% até 89%. Um *framework* de programação, disponibilizado como software livre, foi desenvolvido para executar as simulações e representa uma contribuição secundária da dissertação.

Palavras-chave: Eficiência Energética, Virtualização, Computação em Nuvem, Alocação de Recursos.

Abstract

DE LA FUENTE VIGLIOTTI, A. P. M.**Energy-Efficient Virtual Machines Placement**. 2015. 91 p. Dissertation (Master) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2015.

Data centers' electric power consumption corresponds to almost 1.5% of the total world wide electric power consumption, with the consequent greenhouse effect and CO2 footprints. Virtualization techniques, which are already used by cloud computing providers, improve the energetic efficiency of data centers infrastructure, since they enable the sharing of a same physical hardware among several Virtual Machines (VMs). An efficient VMs placement can reduce the hardware and energy needs.

Most of the existing VMs placement algorithms focuses on the sharing of a single resource, like the processor, or assumes that resources demands are deterministic. In this dissertation, algorithms to place VMs, aiming the reduction of electric energy, are presented and compared. Besides, the algorithms consider multiple stochastic resources. The algorithms implement VMs consolidation mechanisms to reduce energy consumption in data centers and without violating SLAs. three algorithms are presented. The algorithms differ by the implemented heuristic. Two are based on the knapsack problem and one models the problem using evolutionary computation. In average, in simulation experiments considering configurations of real computers, the proposed algorithms reduced the energy consumption starting from 52% up to 89%. A development framework, made available as free software, was developed to run the simulations and represents a secondary contribution of the dissertation.

Keywords: Energy Efficiency, Virtualization, Cloud Computing, Resource Allocation.

Contents

List of Abbreviations

CSV	*Comma Separated Values* (Valores separados por vírgula)
OS	*Operating System* (Sistema operacional)
SLA	*Service Level Agreement* (Contrato de nível de serviço)
VM	*Virtual Machine* (Máquina virtual)
VMs	*Virtual Machines* (Máquinas virtuais)

List of Figures

List of Tables

Chapter 1

Introduction

Energy efficiency has become one of the most important design requirements for computing systems such as data centers and cloud systems, especially in this era of climate change and global warming. Energy consumption by data centers represented between 1.1% and 1.5% of the total world wide electric power consumption in 2010 Koomey (2011). This corresponds to the typical yearly electricity consumption of 120 million households, producing negative greenhouse effects and CO_2 footprints, and worst of all, it is growing rapidly Koomey (2011). A way to help with this issue is by finding energy-efficient techniques and algorithms to manage computing resources Beloglazov *et al.* (2010) Fettweis e Zimmermann (2008).

It is important to observe that the performance per watt ratio of computers has been constantly increasing every year, nevertheless the power consumption keeps increasing as well. If this trend continues, the cost of the energy consumed by a server during its lifetime will exceed the hardware cost. The problem is even worse for clusters, data centers and large-scale computing infrastructures. An energy consumption rise of 16-20% per year can be observed in the last years, corresponding to a doubling every 4-5 years

Increased flexibility and improved efficiency can be achieved by using virtualized systems which is the cornerstone technology that makes cloud computing and utility computing a fact nowadays. The virtualization technology allows the consolidation, also known as allocation, of several servers (Virtual Machines or VMs) into one physical machine reducing the hardware in use. It allows new business models based on pay-as-you-go basis reducing maintenance of own computing environment costs.

Energy efficient VM allocation takes advantage of the fact that hardware can be switched to low-power modes to reduce the overall energy consumption. Later it can be reactivated when an increase of resources use is detected.

1.1 Objectives

One option to reduce energy and achieve more benefits of VM consolidation is to oversubscribe the resources available. For instance, on the OpenStack, a software used to manage virtual machines, it is possible to increase the over-commit ratio OpenStack Manual - Chapter 5 - Scaling . By doing this, it is possible to get higher CPU utilization levels, however if the CPU is aggressively oversubscribed it could lead to performance degradation.

It is common to estimate the power consumed by a host based on the CPU utilization. While

CPU utilization based models are able to provide an accurate prediction for CPU-intensive applications, they tend to be inaccurate for other types of applications like RAM memory or I/O intensive applications, as concluded by Dhiman *et al.* (2010).

Several related works propose mechanisms to allocate VMs with the objective to reduce energy consumption, as Beloglazov e Buyya (2012b) Beloglazov e Buyya (2013) Meng *et al.* (2010). However, the common assumption of these studies is that, either they observe just one resource (for example, CPU) or they consider that all resources are deterministic (all the resources demands are stable over time). Such assumption is not true all the time as shown by Chen *et al.* (2011), since some resources are allocated with stochastic demands which are difficult to accurately estimate.

To evaluate mechanisms to allocate VMs it is needed to run them in real data centers or in simulators with realistic workloads. Simulation is an attractive option when one does not have access to datacenters to evaluate his/her proposal. Cloud computing simulators like CloudSim Rajkumar Buyya (2013) and GridSim GridSim make their decision only based on CPU observation, or do not consider power efficiency VMs placement. So, a new cloud computing simulator considering these features represents an important contribution.

Hereupon, the objectives of this dissertation are:

- To study the power savings impact of VM consolidation using a knapsack based algorithm.
- To propose an heuristic based on a knapsack based algorithm.
- To study the power savings impact of VM consolidation using a genetic algorithm.
- To propose an heuristic based on a genetic algorithm.
- To study the impact of using several resources in the VM consolidation process.
- To build a cloud simulation framework to simulate the proposed algorithms.

1.2 Contributions

The contributions of this dissertation can be divided into 3 categories: design of novel algorithms for VMs allocation, aiming energy savings, considering multiple resources, design of novel heuristics for non-linear power consumption models, and implementation of a framework for simulation of energy-efficient VM allocation algorithms. The detailed contributions are:

1. Three novel algorithms for VMs allocation considering multiple resources. Two using a knapsack model and one using an evolutionary computation model. The objective of these algorithms is to reduce the energy consumption without violating SLAs.
2. Two novel heuristics for non-linear power consumption models using the real power consumption models provided by the SPECpower benchmark.
3. Software implementation of pyCloudSim [1], a framework for simulation of VMs allocation algorithms:

[1]https://pycloudsim.albertdelafuente.com . Last access on January 15, 2015

- An open source software implementation of the framework in Python released under the Apache 2.0 license and publicly available online.
- An implementation of several algorithms for VMs allocation.

During the writing of this dissertation the contributions were published in three conference papers:

- Albert De La Fuente Vigliotti, Daniel Macêdo Batista, “Energy-Efficient Virtual Machines Placement", Proceedings of the 2014 Brazilian Symposium on Computer Networks and Distributed Systems (SBRC), pages 1–8, 2014. Vigliotti e Batista (2014)
- Albert De La Fuente Vigliotti, Daniel Macêdo Batista, “A Green Network-Aware VMs Placement Mechanism", Proceedings of the IEEE Global Communications Conference (GLOBECOM), pages 1–6 , 2014. De La Fuente Vigliotti e Batista (2014)
- Thiago Kenji Okada, Albert De La Fuente Vigliotti, Daniel Macêdo Batista, Alfredo Goldman vel Lejbman, “Consolidation of VMs to improve energy efficiency in cloud computing environments", Proceedings of the 2015 Brazilian Symposium on Computer Networks and Distributed Systems (SBRC), pending for publication.

1.3 Organization

The reminder of the dissertation is organized as follows:

- Chapter 2 presents the concepts used on this work as well as the scope of this dissertation and its positioning in the area.
- Chapter 3 presents an overview of the state of the art literature on energy-efficient VMs allocation.
- Chapter 4 presents the proposed algorithms to allocate VMs aiming energy efficiency and considering multiple resources.
- Chapter 5 describes the architecture and implementation of pyCloudSim, a simulation framework for energy-efficient VMs allocation algorithms. The results are also presented and discussed, as the experiments performed to evaluate the performance of the algorithms.
- Chapter 6 concludes the dissertation with a summary of the findings and discussion of future research directions.

Chapter 2

Concepts

This chapter presents the concepts needed to understand the contributions of this dissertation. On Section 2.1 the server virtualization concept will be introduced, which is the cornerstone of Cloud Computing. The Cloud Computing concept will be introduced on Section 2.2, and the Service Level Agreement concept on Section 2.3. On Section 2.4 an explanation about energy-efficiency on computing systems is made, and some power models are reviewed on Section 2.5. On Section 2.7 the mathematical optimization techniques used in this work are detailed.

2.1 Virtualization

In this section the server virtualization concept is presented. There are other types of virtualization that are out of the scope of this study, like network virtualization, memory virtualization, and database virtualization for example. Server virtualization is a technology that allows to create logical computing units called Virtual Machines (VMs) that can accommodate an individual OS. These VMs are able to run applications transparently isolated from the physical hosts where the VMs run.

Although the virtualization term is being widely disseminated nowadays, it first appeared with the IBM mainframes in the 1960's, as described in the work by Creasy (1981). At that time, the virtualization was used to share system resources, including I/O and some privileged operations. Nevertheless, it was commercialized for the x86-compatible computers only in the 1990's Beloglazov *et al.* (2010). Virtualization has become broadly available as hardware performance increased over time. Currently, it is supported natively on most modern architectures like Intel Intel-VT and AMD AMD-V . Several commercial companies VMware Inc. OpenVZ project and open-source projects Kernel Based Virtual Machine Oracle VM VirtualBox offer software packages to allow the use of virtualization.

Classic benefits of virtualization include improved hardware utilization, manageability and reliability. Several VMs with different requirements can be hosted on the same server. With this isolation abstraction it is possible to achieve a high granularity to such an extent it has services isolated into separated VMs. This allows to schedule OS and software upgrades reducing the impact of downtime and failures.

Virtualization also allows to improve system security by isolating multiple applications into their own VMs. Intrusions would be confined to the VM in which they occur. It is possible to enhance reliability as software failures in one VM will not affect other VMs running within the same host

Uhlig *et al.* (2005).

Data centers take great advantage of virtualization, since this technology is able to improve the utilization of server resources, and thus, to reduce power consumption and electricity bills. The benefits of virtualization scaled to a data center level have enabled the use of thin clients, by allowing multiple users to connect to a pool of servers over the network. This practice could be used as part of a Green IT policy, considering that thin clients consume significantly less energy compared to a regular workstation. Thin clients started gaining relevance with the adoption of Software as a Service (SaaS), or Virtual Desktop Infrastructures (VDI) such as VMware View and Citrix Xen-Desktop Beloglazov *et al.* (2010).

2.1.1 Hypervisor based Virtualization

Hypervisor based virtualization prioritizes isolation of resources, libraries and Operating System (OS) from each VM from the host over sharing, by running a full OS both on the virtual machine, also known as a guest machine or only guest, and on the host, with the consequent overhead. Some examples of software that provide a hypervisor based virtualization are VMware VMware Inc. , Xen Xen Project and KVM Kernel Based Virtual Machine . This type of virtualization is called "hypervisor based" because the OS, usually directly installed on the host, that control the guests is called hypervisor. The name Virtual Machine Manager (VMM) is also used to refer to the hypervisor.

As technology evolved and many hardware virtualization extensions like Intel-VT Intel-VT and AMD-V AMD-V got better over time, there has been a noticeable acceleration of key functions of the virtualized platform. This efficiency offers benefits, including: (A) speeding up the transfer of platform control between the guest OSs and the hypervisor, (B) enabling the hypervisor to uniquely assign I/O devices to guest OSs and (C) optimizing the network for virtualization with adapter-based acceleration. Nevertheless there is still a performance gap, specially regarding I/O operations, since some tasks need to be intermediated by the hypervisor.

2.1.2 Operating System based Virtualization

Operating system based virtualization, also know as containers, Virtual Private Servers or jails is a virtualization method where the kernel of an operating system allows to have multiple isolated user space instances rather than just one. Some examples of software that provide an operating system based virtualization are LXC
LXC Development Group , CGroups Jackson e Lameter (2013), OpenVZ OpenVZ project and Linux-VServer Linux-VServer Development Group . The terms "containers" and "jails" are used to refer to each isolated user space instance.

The main advantage of operating system based virtualization is that this method allows to share libraries from the guest and host OS to improve the performance and reduce the virtualization overhead. Other typical scenarios include application separation to improve security and migration of containers for load balancing in a cluster.

Recent studies have shown how operating system based virtualization could be used even in High Performance Computing (HPC) environments, as shown by Xavier *et al.* (2013), where they have conducted a number of experiments to perform an in-depth performance evaluation of container-based virtualization for HPC. Such experiments included several performance benchmarks comparing native performance, container-based performance and hypervisor based virtualization such as Xen. In their experiments they have showed that container based virtualization performance is better than hypervisor based virtualization and, in some cases, almost similar to the native performance. The problems when using hypervisor based virtualization technology in HPC environments are caused mainly because of the high overhead of having a second operating system. The two virtualization approaches share advantages, like migration, which allows to move a VM to another host and fail over, which allows a VM to continue operations if a host fails, but some features are unique to each virtualization platform, like full network virtualization in hypervisors and performance in containers.

Operating system based virtualization is not as flexible as a hypervisor based virtualization solution since the guests cannot run a different operating system from the main host, for example, a Linux host is not able to have Windows as a guest. However, it is possible to guest different Linux distributions.

It is important to observe that both of the virtualization techniques allow to migrate VMs from one physical host to another physical host using live or off-line migrations.

2.2 Cloud Computing

Cloud computing refers to both the applications delivered as services over the Internet and the hardware and software in the data centers that provide those services. This technology heavily relies on server virtualization.

There are many definitions for *Cloud computing*, often related to cloud and web services for marketing reasons, however cloud computing is more than that. The most common adopted definition is made by the *National Institute of Standards and Technology (NIST)* Mell e Grance (2011):

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.". The cloud model is composed of the following five essential characteristics:

- On-demand self-service. A consumer can provision automatically computing capabilities, such as server time and storage. No human intervention is required.
- Broad network access. Capabilities are available over the network to promote use by heterogeneous thin or thick client platforms (mobile phones, tablets, laptops and workstations)
- Rapid elasticity. Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly in response to changes in the demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.
- Measured service. Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service. Resource usage can be monitored, controlled, and reported , providing transparency for both the provider and consumer of the utilized service.

Cloud computing has many benefits, among them the most important to be highlighted are: *costs savings* – organizations no longer have to invest money in their own IT infrastructure, as services are available on demand paid as they are used, so there are decreasing investments and running costs; *efficiency* – cloud providers can operate better, faster and in a more cost-efficiently way than own IT infrastructure; *improve availability* – it is common to find load balancing, redundant connections and state of the art infrastructure setup to meet business needs. It also improves *scalability* – services can be scaled for business needs as resources can be allocated dynamically Brian *et al.* (2008).

Mell e Grance (2011) have also classified the service models into three groups:

- Software as a Service (SaaS). The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface or a web browser. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application settings. Examples of SaaS are: Google's Gmail and Facebook.

- Platform as a Service (PaaS). The capability provided to the consumer is to deploy, onto the cloud infrastructure, consumer-created applications using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure but has control over the deployed applications and possibly settings for the application-hosting environment. Examples of PaaS are: Heroku and Google Apps.

- Infrastructure as a Service (IaaS). The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software (OSs and applications). The consumer does not manage or control the underlying cloud infrastructure but has control over OSs, storage, and deployed applications; and possibly limited control of select networking components like firewalls. Examples of IaaS are: Amazon Web Services and Rackspace.

Cloud computing has changed a large part of the IT industry. Innovative ideas for new Internet services no longer require large investments to start. The companies no longer need to be concerned about over-provisioning for a service nor under-provisioning since they can easily scale their infrastructure to meet the demands as they grow. It is also possible for companies with large tasks to get results quickly by scaling their programs into more servers, since using 1000 servers for one hour costs the same as using one server for 1000 hours. Soltesz *et al.* (2007)

2.3 Service Level Agreement

A Service Level Agreement (SLA) can be defined as a formal negotiated agreement between two parties, by covering many aspects of the relationship between the service provider and the consumer Trygar e Bain (2005). The most notable aspects are related to services performance, customer support, service provisioning and billing.

The main objective of an SLA is to establish the Quality of Service (QoS) constraints, priorities and responsibilities. Cloud computing SLAs can be described as minimum allowed throughput, maximum response time, or latency delivered by the deployed system. One of the important requirements for a Cloud computing environment is providing reliable QoS, otherwise the Cloud provider often refunds the consumer in case of SLA violation as a penalty.

When proposing energy-efficient VMs allocation, one needs to be aware of the SLA to avoid performance degradation of the consumer applications, including increased response times, time-outs or even failures. This can happen because sometimes, when only the energy consumption is considered, the best solution can be to consolidate all VMs into one physical machine, which can cause SLA violations of performance limitation of this machine. Therefore, Cloud providers have to establish QoS requirements to avoid SLA violations and deal with the energy-performance trade-off - meeting the QoS requirements while minimizing energy consumption Beloglazov *et al.* (2010).

2.4 Energy Efficient Computing Systems

In 2010 it was estimated that the energy consumption by IT infrastructures would be accounted between 1.1% and 1.5% of the global electricity use as showed by Koomey (2008). This results in substantial carbon dioxide (CO_2) emissions, which were estimated to be 2% of the global emissions according to Gartner, Inc. .

Along with hardware design, energy efficiency is influenced on how the software manages the resources. Energy-efficient resource management techniques were first introduced on mobile devices, where it has direct impact on battery lifetime Beloglazov *et al.* (2010). These techniques can be adapted for servers and data centers.

According to Feng (2014), the infrastructure equipment related to power and cooling may be responsible for about half of total annualized costs in a data center facility. This fraction is growing over time as energy use increases and IT equipment costs decrease.

The author in Feng (2014) discusses how an aggregate parameter is affecting the cost of data center. This parameter is the amount of direct power use (Watts) associated with $1000 of expenditure on servers, which has been increasing rapidly over the last years. The fact has made cooling and power infrastructure costs a challenge in the construction of data centers. If this trend continues, the costs related to powering a data center will exceed the IT capital costs in under a decade.

For example, at 100 Watts per $1000 of server costs, the IT capital costs represent about 40% of the total costs. At 200 Watts per $1000, they represent less than one third of total costs. This means that for every $1 spent on IT equipment a company would be paying at least $2 for electricity use, power cooling and other costs. This relationship allow to understand the underlying effect of power per server cost in an unambiguous way.

After analyzing several server specifications from 2000 to 2009, Feng (2014) has concluded that performance trends for server systems seemed to follow the popular Moore's law by doubling every 1.5 to 2 years in most cases. Performance per server cost increased more rapidly than performance per Watt, which drives power use per server cost up over time.

2.4.1 Energy and Power Models

To understand power and energy management mechanisms it is necessary to introduce some concepts. *Electric current* is the flow of electric charge, measured in amperes (A). *Amperes* define the amount of electric charge transferred into a circuit per second. Power and energy can be defined according to the work that a system performs. *Power* is the rate at which the system performs the work, measured in watts (W). *Energy* on the other hand is the total amount of work performed on a period of time, measured, for instance, in watt-hour (Wh). The *potential difference* is measured in volts (V). The work is done at the rate of 1W when 1A is transferred through a potential difference of 1V.

A reduction of power consumption does not always reduce the consumed energy. For example, power consumption can be decreased by lowering the CPU performance. However, if a program takes longer to complete its execution, it may consume the same amount of energy, or even more. Decreased energy consumption reduces electricity bills since the power companies charge by the energy consumption Beloglazov *et al.* (2010).

The lowering and the rise of the CPU performance can be performed by the Dynamic Voltage and Frequency Scaling (DVFS) mechanism, a mechanism present in modern processors that consists of the combined change of the supply voltage and clock frequency. The main idea is to decrease the voltage and the frequency of the CPU by scaling it down when it is not fully utilized and do the opposite actions when it is being entirely utilized.

2.4.2 Sources of Power Consumption

As a consequence of the increasing improvements of energy efficiency on CPUs, with mechanisms like DVFS, the CPU no longer dominates the power consumption on a server. While DVFS optimizes the CPU consumption to consume less than 30% in their peak power in low-activity modes Barroso e Holzle (2007), saving more than 70%, the dynamic ranges of power savings in other components are much narrower: less than 50% for Dynamic Random Access Memory (DRAM), 25% for disk drives and negligible for other components Fan *et al.* (2007).

With the growing amount of memory in modern servers, the power consumption by the memory is becoming higher than the power consumed by the CPU. For instance, the power consumption by a server with eight 1GB DIMMs is about 80W. Modern large servers currently use 32 or 64 DIMMs. This fact makes memory one of the most important server components that has to be efficiently managed nowadays. However, most power management techniques are still focused on the CPU.

Power supplies also have significant power losses due to the inefficiency of the current technology. They achieve the highest efficiency at loads within the range of 50-75%. However, most data centers normally have a load of 10-15%, wasting the majority of the consumed electricity and leading to the average power losses of 60-80%.

2.5 Power Models

To be able to propose energy-efficient VM allocation algorithms it is important to predict the power consumption of the physical machines as a function of their utilization.Fan *et al.* (2007) have proposed both a linear model based on a CPU utilization and a nonlinear model using a calibration parameter that minimizes the square error obtained experimentally.

The linear model proposes that the power consumption by a server grows linearly related to the growth of the CPU utilization, starting from the power consumption in the idle state up to the power consumed when the server is fully utilized. This relationship can be expressed as shown in Equation 2.1:

$$P(u) = P_{idle} + (P_{busy} - P_{idle})u, \tag{2.1}$$

where $u \in [0,1]$ is the current CPU utilization, P is the estimated power consumption, P_{idle} is the power consumed by an idle server ($u = 0$), and P_{busy} is the power consumed by the server when it is fully utilized ($u = 1$).

The empirical nonlinear model proposed is given in Equation 2.2:

$$P(u) = P_{idle} + (P_{busy} - P_{idle})(2u - u^r), \tag{2.2}$$

where r is an experimentally obtained parameter that must be calibrated for each class of machines. The idea behind this model is to minimize the square error.

There are other models available on the literature like Jaiantilal *et al.* (2010). However, this model is dependent on low level hardware details like processor cycles which makes it hard to use.

2.5.1 SPEC Power Benchmark

The Standard Performance Evaluation Corporation (SPEC) Standard Performance Evaluation Corporation (a) was formed by computer industry participants in 1988 to establish industry standards for measuring computing performance. SPEC has grown to more than 60 member companies and it became one of the most successful performance standardization references.

In response to the increasing demand of power and performance benchmarks on energy-efficient IT equipment, the SPEC community with the help of leading engineers and scientist in the field of energy efficiency created SPECpower, an initiative to standardize power benchmarks measurement.

SPECpower proposes useful metrics to analyze the newest generation of IT equipment. In Standard Performance Evaluation Corporation (a) it is possible to find more than 20,000 peer-reviewed performance reports.

In December of 2007, the *SPECpower_ssj2008* Standard Performance Evaluation Corporation (b) was released to measure and benchmark the performance characteristics of server-class compute equipment by considering CPUs, caches, memory hierarchy and the scalability of shared memory processors on multiple utilization levels. It uses a power analyzer and temperature sensor daemon, also known as *PTDaemon* which presents a common TCP-IP based interface to hide the details of different power analyzer interface protocols Huppler *et al.* (2012).

Performance benchmarks heavily relies on a business model. For example, some environments have application servers that require minimal memory and low networking capabilities, while others may store large amounts of data on disk, requiring also large amounts of memory to process the information. Some environments might need heavy load on all the processors, as well as other resources.

It is not possible for a single benchmark to represent the power efficiency for all the possible combinations of an IT environment, nevertheless power characteristics of environments that are very closely replicated by the configuration of the benchmark can be accurately predicted. However it is very possible for a single benchmark to give an indication of the relationship between power and performance for the components that the benchmark stresses Lange *et al.* (2012).

It is important to measure at least the power used on maximum performance and the power used during the active-idle period, where the system is ready to work, however without any workload request for some period of time. These are the best and worst cases for work per unit of power. This is the idea behind most linear models and the model proposed by Fan *et al.* (2007), where these best and worst cases are used as high and low bounds to define the power characteristics, and any utilization threshold in between can be predicted linearly.

Several environmental consideration are also considered on the benchmarks like:

- Temperature, which substantially affects the power characteristics. Generally lower temperature from the air conditioning ensures lower power operations. Traditional air cooled equipment is assumed, even with the authors of the benchmarks discussing that alternative cooling systems, like liquid cooling, are more power-efficient. They define a minimum acceptable temperature of 20 degrees Celsius.

- Air pressure and altitude, which can affect the density of the air used to cool the equipment. They consider a safe maximum of 1050 mill bars as a reasonable natural atmospheric condition.

- Humidity, which they consider to have an almost negligible impact on the power characteristics, and therefore no boundaries for relative humidity are setup.

- Air flow, which generally does not affect unless in extreme conditions.

- Power source, which is the line voltage and other characteristics may affect the efficiency. The considered conditions specifications are:

  - Frequency: (50Hz, 60Hz) $\pm 1\%$
  - Voltage: (100V, 110V, 120V, 208V, 220V, 230V or 400V) $\pm 5\%$

The SPECpower benchmark is able to deliver a specific throughput result to adjust to a variety of load levels. The proposed method is to measure the power consumption at 11 measurement intervals: $100\%, 90\%, 80\% \ldots 20\%, 10\%$ and active-idle, represented by a 0% throughput measurement.

An example of a result after running the SPECpower benchmark is shown on Figure 2.1. The red bars represent the efficiency (throughput per Watt) for each measurement interval. The measurement is performed in terms of the server side Java operations (or ssj_ops) per second per watt. The blue lines shows the average power requirement at each of the 11 measurement intervals. Active-idle is also included without throughput. This benchmark is from a Fujitsu Server PRIMERGY RX1330 M1.

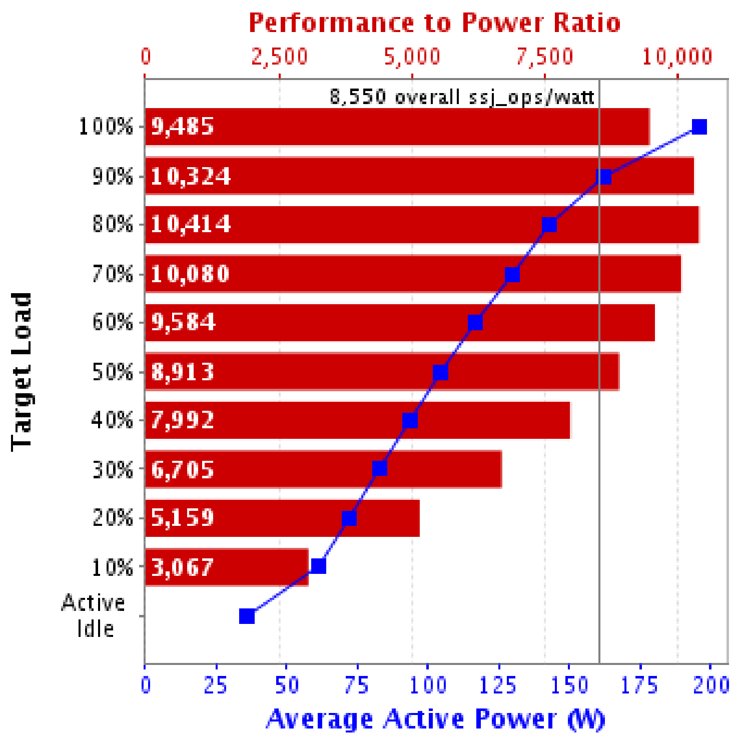


Figure 2.1: *SPECpower measurement example of a Fujitsu PRIMERGY TX2540 M1 server SPECPower Fujitsu PRIMERGY RX1330 Server*

2.6 Energy Efficient Virtual Machines Placement

The problem of energy consumption minimization under constraints is too complex to be treated analytically since it is an NP-hard problem Jung *et al.* (2010). Allocation of VMs can be divided into two problems, the first one is the placement of a new VM on a host while the second one is the optimization of the current VMs allocation.

Energy efficient VM allocation takes advantage of the fact that idle hosts can be suspended to low-power modes to reduce the overall energy consumption. Later, the hosts can be reactivated, for example remotely over the network by using Wake-on-LAN
IBM Announces Universal Management - Wake on LAN when an increase of resources use is detected. It is important to carefully evaluate where the consolidation of the VMs is going to take place, to avoid the migration of VMs to an overloaded host and to fall into an SLA violation.

The process to allocate virtual machines considered in this work is based on the work proposed by Beloglazov e Buyya (2012a). This process consists in performing periodical dynamic consolidation of VMs by packing them on as fewer physical machines as possible to conserve energy in virtualized data centers. The reason behind this idea is that several modern computers are more energy-efficient when they are operating nearly their full capacities Beloglazov e Buyya (2012a). This approach can be divided into the following sub-problems.

- **Information retrieval**. Information gathering about the VMs resource consumption (CPU, network, memory, I/O, etc)

- **Underload detection**. It's important to detect hosts that can be considered underloaded because they will be operating with a bad efficiency. In this case all the VMs from this host should be migrated to another host (if possible) and later the “old host” should be switched to a low-power mode.

- **Overload detection**. Similar to the above step, to ensure Quality of Service (QoS) requirements, if a host is considered overloaded, some VMs should be migrated to another host.
- **VMs selection**. Select which VMs to migrate from an overloaded host.
- **VMs migration**. Migrate the selected VMs on other hosts.

2.7 Optimization Techniques

Since the problem to solve is NP-hard, it is important to review some mathematical optimization modeling concepts and approaches. They are used in the algorithms on Chapter 4.

2.7.1 Linear Programming

Linear Programming (LP) is a method to optimize (minimize or maximize) a linear function while satisfying a set of linear equality and/or inequality constraints or restrictions.

The linear programming problem was first conceived by Leonid Kantorovich in 1939 during the second World War as a way to plan expenditure, returns and to reduce costs of the army Bazaraa *et al.* (2009). Kantorovich work remained unknown until the end of the war.

During and after World War II, it became evident that planning and coordinating the efficient utilization of resources were very important. Intensive work by the United States Air Force team SCOOP (Scientific Computation of Optimum Programs) began in June 1947. As a result, the simplex method was developed by George B. Dantzig by the end of the summer of 1947. In 1948 George B. Dantzig published a paper Bazaraa *et al.* (2009) where he addressed this method as "Programming in a Linear Structure", hence the conception of linear programming problems is usually credited to him. At that time he was working as a mathematical advisor of the United States Air Force Comptroller at the Pentagon, developing a mechanized planning tool for a time-staged deployment, training, and logistical supply program Bazaraa *et al.* (2009).

Later in 1949 Dantzig published the “simplex method" for solving linear programs. The simplex method had wide acceptance because of (1) its ability to model important and complex management decision problems, and (2) its capability for producing solutions in a reasonable amount of time.

A linear programming problem can be expressed with the following two parts:

1. A linear function to be maximized: $f(x_1, x_2) = c_1x_1 + c_2x_2$

2. And the problem constraints on the following form:

$$a_{11}x_1 + a_{12}x_2 \leq b_1$$
$$a_{21}x_1 + a_{22}x_2 \leq b_2$$
$$a_{31}x_1 + a_{32}x_2 \leq b_3$$

Or in its canonical form:

$$\begin{aligned} \text{Maximize:} \quad & c^T x \\ \text{Subject to:} \quad & Ax \le b \\ \text{and} \quad & x \ \ \ge 0 \end{aligned}$$

The solution will be the values of x that maximize the objective function.

One of the main advantages of LP problems is that if they are small enough they can be represented graphically. For instance, the problem shown from Equation (2.3) to Equation (2.6) represents a linear programming problem. The problem can be represented graphically as Figure 2.2, where the Equation (2.3) represents the objective function (thick black line). The Equation (2.4) represents the first constraint of the problem (shown as the area defined below the red line). The Equation (2.5) represents the second constraint of the problem (shown as the area defined below the green line). The Equation (2.6) represents the third $x_1 \ge 0$ (shown as the area above the purple line) and the fourth constraint $x_2 \ge 0$ (shown as the area on the right of the blue line).

$$\begin{aligned} \text{Maximize:} \quad z = \quad & 4x_1 + 3x_2 && (2.3) \\ \text{Subject to:} \quad & 2x_1 + \ \ x_2 \le 4 && (2.4) \\ & -3x_1 + 2x_2 \le 3 && (2.5) \\ \text{and} \quad x_1, x_2 & \ge 0 && (2.6) \end{aligned}$$

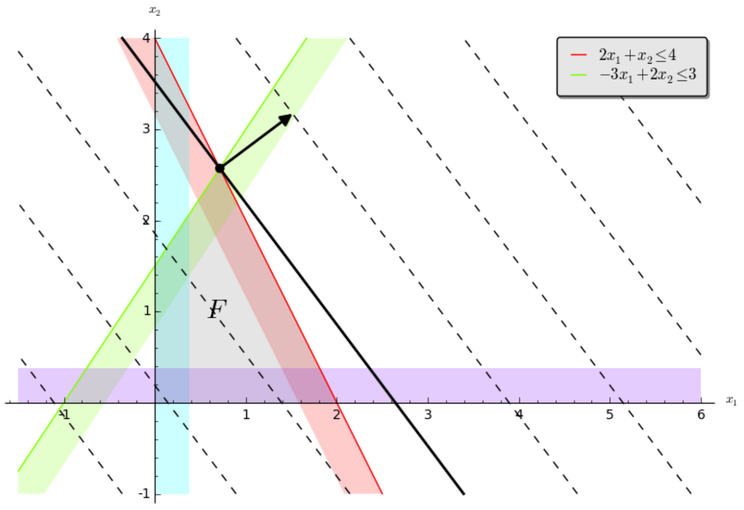


Figure 2.2: *Linear programming example*

The intersection of all the regions is known as the feasible region. The parallel dotted lines represents an idea of how the objective function behaves. The intersection of the line within the feasible region and the maximum possible objective function is the best solution. In this case $x_1 = \frac{5}{7}$ and $x_2 = \frac{18}{7}$.

When the Simplex method was first introduced by Dantzig, the research community intuition was that this algorithm would not be very efficient. The idea behind the Simplex method is to travel the edges of a polyhedron without making any attempt to skip through some of them. However, researchers were pleasantly surprised when, in practice, this method performed very well. For most practical problems, this method has an empirically complexity of roughly $3m/2$ iterations, and seldom more than $3m$ iterations, being m the number of unknown variables. This means that the Simplex method has a polynomial-time average-case complexity. Nevertheless in 1971, Victor Klee and George Minty produced a class of problems in which the Simplex method requires $2^n - 1$ iterations which lead to an exponential worst-case complexity Bazaraa *et al.* (2009).

2.7.2 The Bin Packing Problem

The one dimension Bin Packing Problem (BPP) is similar to the Knapsack Problem (KSP) considering multiple knapsacks. The idea is to pack different items into a finite number of bins, each of the bins has a size restriction. The objective is to minimize the number of bins used. Several variations of the BPP can be found like 2D BPP and 3D BPP and they have applications for instance in filling containers or loading trucks with weight or volume constraints.

The BPP is an NP-complete problem. In fact, it is strong NP-complete as shown by Garey e Johnson (1978). The one dimension BPP can be defined as follows:

Given a set of bins S of size V and a list of n items with sizes $a_1, \ldots, a_n$ to pack, find an integer number of bins B and a B-partition $S_1 \cup \cdots \cup S_B$ of the set $\{1, \ldots, n\}$ such that $\sum_{i \in S_k} a_i \leq V$ for all $k = 1, \ldots, B$. A solution is optimal if it has minimal B.

A possible Integer Linear Programming formulation of the problem is:

$$\begin{aligned}
\text{Minimize:} \quad & B = \sum_{i=1}^{n} y_i \\
\text{Subject to:} \quad & \sum_{j=1}^{n} a_j x_{ij} \leq V y_i, && \forall i \in \{1, \ldots, n\} \\
& \sum_{i=1}^{n} x_{ij} = 1, && \forall j \in \{1, \ldots, n\} \\
& y_i \in \{0, 1\}, && \forall i \in \{1, \ldots, n\} \\
& x_{ij} \in \{0, 1\}, && \forall i \in \{1, \ldots, n\} \, \forall j \in \{1, \ldots, n\}
\end{aligned}$$

where $y_i = 1$ if bin i is used and $x_{ij} = 1$ if item j is put into bin i Martello e Toth (1990).

It is possible to model the energy-efficient VM placement as the problem known as VM packing. In this case the items are the VMs to be packed on each bin, and each bin is a server that will host a VM. Besides, the memory requirements of the VMs allocated into the same server can be reduced by sharing some memory pages among the VMs as shown by Sindelar *et al.* (2011). This type of BPP is even hard to approximate.

The energy-efficient VM placement problem has been first modeled as an instance of the BPP problem. Unfortunately the initial algorithm could not scale for a large number of VMs and hosts for its natural complexity and therefore the need of heuristics to solve the problem. More details about the algorithms are presented on Chapter 4.

2.7.3 Genetic Algorithms

In 1957 Alex Fraser started researching on simulations of artificial selection of organisms Fraser (1960). Fraser's simulations included the essential elements of modern genetic algorithms. Nevertheless, Genetic Algorithms (GAs) became popular through the work of John Holland in the early 1970's with his book *Adaptation in Natural and Artificial Systems*. Holland's work, his colleagues and his students at the University of Michigan Holland (1975) originated with the studies of cellular automata. The goal of their research has been twofold: (1) to abstract and rigorously explain the adaptive processes of natural systems, and (2) to design artificial systems software with the mechanisms of natural systems Goldberg (1989).

Nowadays genetic algorithms have widespread applications in business, scientific and engineering problems. The main reason is that they are not restricted to assumptions about the search space like continuity or the existence of derivatives like gradient-based optimization methods.

Before continuing with this section it is important to review the common terms used in this area. A detailed introduction to genetic algorithms, and other metaheuristics, can be found on the book *Essential of Metaheuristics* Luke (2013).

- Individual: a candidate solution.
- Child and parent: a child is a modified version of a candidate solution (parent).
- Population: a set of candidate solutions.
- Fitness: a quality quantification of a solution.
- Fitness landscape: a quality function.
- Evaluation: computing the fitness of an individual.
- Selection: picking individuals based on their fitness.
- Mutation: a plain small, bounded but random modification of a solution. This is often thought as a "asexual" breeding.
- Crossover: a special modification of a solution which takes two parents to swap sections of them. This is often thought as a "sexual" breeding. Also might be referred as Variators.
- Breeding: to produce one or more children from a population of parents through an iterated process of selection and introducing a small, bounded but random change (typically mutation or recombination)
- Genome: an individual's data structure, as used during breeding.
- Chromosome: a genome in the form of a fixed length vector.
- Gene: a particular slot position in a chromosome.
- Allele: a particular setting of a gene.
- Phenotype: how the individual operates during fitness assessment.

- Generation: the population produced by each cycle (one cycle of fitness assessment, breeding, and population reassembly).

A genetic algorithm iterates through fitness assessment, selection breeding and population reassembly. To breed, an empty population of children is taken at the beginning. Then two parents from the original population are selected, and crossed and the results are mutated. This forms two children, which are then added to the child population. The process is repeated until the child population is entirely filled. The genetic algorithms are classically operated over binary vectors of fixed length.

To perform the breeding a crossover and mutation is required. The *mutation* on a boolean vector can be achieved by using bit-flip mutation, which is equivalent to flip a coin with a certain probability, often $1/l$ where l is the length of the vector. Each time the coin comes heads, a flip on the bit is applied. While there are other types of mutation depending on how the information is represented, the focus will be on the binary representation which is the one used on the algorithms. The objective of the mutation is to avoid local minimums by preventing the population of chromosomes from becoming too similar to each other and thus slowing or even stopping the evolution.

The *crossover* involves mixing and matching parts of two parents to form a child. There are many ways to perform a crossover operation, the most common are *one-point*, *two-point*, *n-point* and *uniform crossover*.

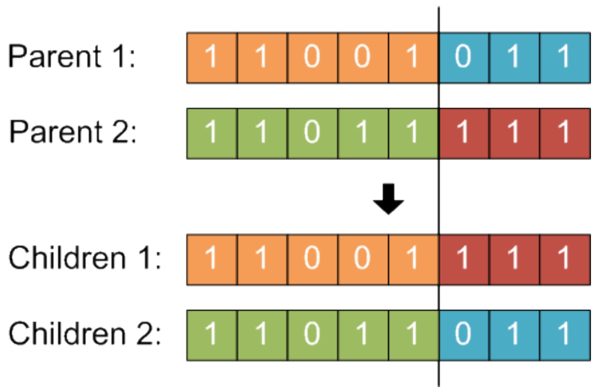


Figure 2.3: *Single-point crossover example ($c = 6$)*

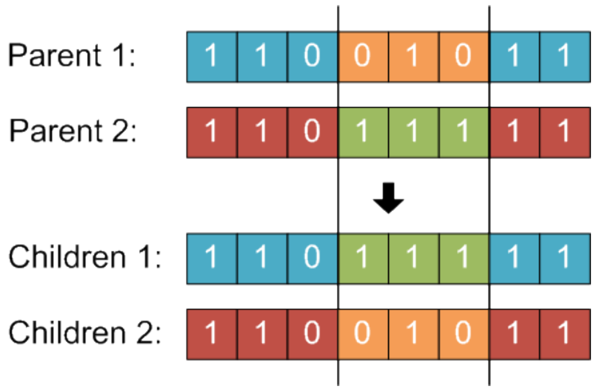


Figure 2.4: *Two-point crossover example ($c = 4$ and $d = 6$)*

The one-point crossover picks a number $c \in \{1, l\}$ and swaps the indexes $i \leq c$, as shown in Figure 2.3. The two-point crossover is similar to the one point except that it defines two cut points

c and d, and it swaps the indexes between them, as shown in Figure 2.4.

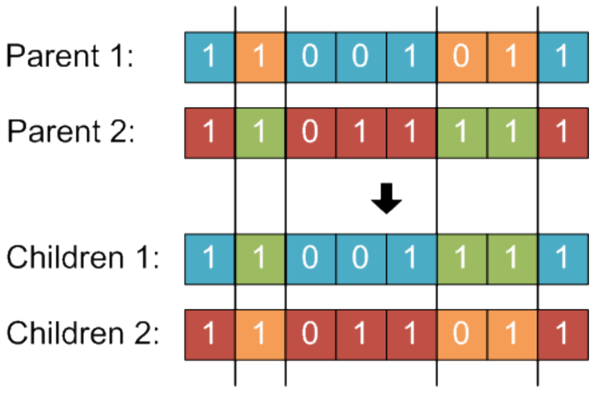


Figure 2.5: *N-point crossover example*

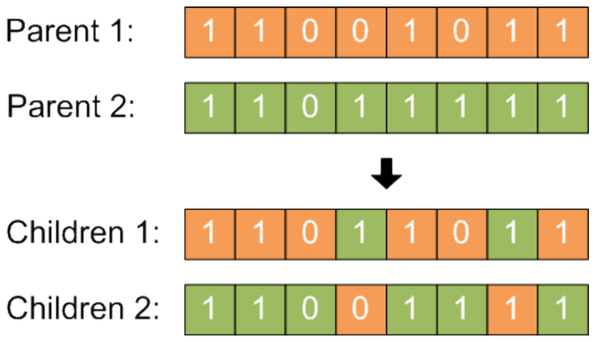


Figure 2.6: *Uniform crossover example*

The n-point crossover might be seen as a generalization of the one-point and the two-point, it chooses n random crossover points and it swaps parts alternating between parents as shown in Figure 2.5. The uniform crossover threats each point independently from any other by swapping individual indexes if a coin flip comes up head with probability p, as shown in Figure 2.6.

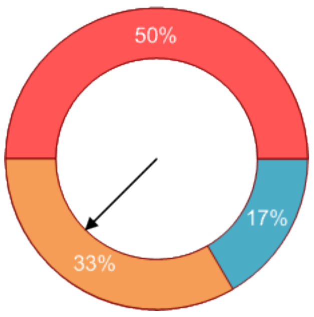


Figure 2.7: *Roulette selection example*

The *selection* is based on the idea that better individuals (higher fitness function value) get higher chances to produce a better solution. There are many selection methods, the most common is the *roulette selection* which selects an individual in proportion to their fitness. The higher the fitness, the higher the probability of that individual to get selected. Graphically is shown on Figure 2.7, where each part of the roulette represents the fitness of an individual. In this case the individual with 50% of the roulette is more probable to be selected.

Sometimes, the fitness function of several individuals might be too similar between several individuals, in this cases the *roulette selection* will be selecting individuals with nearly identical probability which would behave as random selection. To fix this, the technique called *tournament selection* could be used.

The *tournament selection* returns the individual with higher fitness function from some t individuals picked at random positions from the population. This method has become the primary selection technique used for genetic algorithms according to Luke (2013).

Genetic algorithms are a metaheuristic and as such there is no general complexity analysis to all of them. In general, the common terminology used is convergence time Rabinovich e Wigderson (1999), which depends on the mutation and crossover probabilities and the population size. A small mutation rate may lead to genetic drift while a high mutation rate may lead to loss of good solutions. A crossover with high rates may lead to premature convergence of the genetic algorithm.

The generational process is repeated until the *termination condition* has been reached. The most common terminations are: finding a solution that satisfies a criteria, a fixed number of generations and computation time reached.

Chapter 3

Literature Review

Most of the presented literature assumes that the consumption grows linearly in relation to the CPU usage. With modern CPUs and multi-core processors this assumption might not be true all the time. This can be observed on some recent SPEC Power Standard Performance Evaluation Corporation (a) benchmarks.

Beloglazov e Buyya (2012b) have discussed several algorithms for offline and online deterministic VM consolidation problem. They have compared the performance of their algorithms using competitive analysis with simulations. They have compared several algorithms:

- An adaptive utilization threshold: median absolute deviation, which sets upper and lower utilization thresholds for hosts and keeps the total utilization of the CPU by all the VMs between these thresholds. It migrates all the VMs of a host when the CPU utilization falls below the lower threshold and it switches the host to a lower energy state to avoid idle power consumption. When the CPU utilization reaches the upper threshold some VMs have to be migrated to prevent SLA violations.

- An adaptive utilization threshold: interquartile range, which proposes a method for setting an adaptive upper utilization threshold based on the interquartile range (IQR) statistical dispersion.

- Local regression, which is based on the Loess method (or local regression (LR)) whose main idea is fitting simple models to localized subsets of data to build up a curve that approximates the original data.

- Robust local regression, which transforms Loess into an iterative method to avoid the LR algorithm's vulnerability to outliers that can be caused by leptokurtic or heavy-tailed distributions.

They have combined the algorithms with different VM selection policies:

- The minimum migration time policy, which migrates a VM that requires the minimum time to complete a migration estimated as the amount of RAM utilized by the VM divided by the spare network bandwidth available for the host.

- The random selection policy, which selects a VM to be migrated according to a uniformly distributed discrete random variable.

- The maximum correlation policy, which selects the VMs to be migrated that have the highest correlation of the CPU utilization with other VMs, since the higher the correlation between the resource usage by applications running on an oversubscribed server, the higher the probability of the server overloading.

They have used a modification of the Best Fit Decreasing (BFD) algorithm called Power Aware BFD (PABFD), where they sort all the VMs in the decreasing order of their CPU utilization to allocate the next VM to the most power efficient node first (least increase of power consumption caused by the allocation). They have focused on a single resource, the CPU, since it is the most oversubscribed resource. Although it is important to consider the CPU resource, the algorithms presented on this dissertation also consider other resources to place the VMs.

The proposal of Beloglazov e Buyya (2013) reduces energy consumption by maximizing the Quality of Service (QoS). They first formulate the problem and showed that to improve the quality of VM consolidation it is necessary to maximize the mean time between VM migrations. They propose an optimal offline algorithm for host overload detection. Later they propose an online algorithm by applying a Multisize Sliding Window workload estimation technique, which is an extended Sliding Window approach by employing multiple windows with different sizes, where a window to be used is selected dynamically using the information about the previous system state and variances of the estimates obtained from different windows. In Beloglazov e Buyya (2013), the only resource considered is CPU, while our algorithms consider other resources like RAM memory, and I/O.

There is a relationship between the total power consumption by a server and its CPU utilization as observed by Fan *et al.* (2007). Basically their model proposes that power consumption by a server grows linearly in relation to the growth of the CPU utilization. Most literature available tackles energy consumption primarily focusing on only one specific aspect, such as minimizing network consumption or maximizing CPU usage, however some of the following studies consider two resources simultaneously. Although this power model based on CPU was used for the proposed algorithms, the SPECpower benchmarks were also used to simulate power models based on real hardware.

Xu e Fortes (2010) approaches the VM placement problem as a multi-objective optimization problem. A multi-objective fuzzy evaluation is proposed to search for solutions. They focus on CPU, memory and power consumption. They have used a linear power model based on CPU using IBM's advanced management module for an IBM BladeCenter. In our algorithms, power models can be either based purely on CPU or they can be based on a server SPECpower benchmark profile, which allows to use our algorithms on diverse hardware.

Ranjana e Raja (2013) have presented the available strategies of power aware VM placement techniques on a survey. They have discussed the algorithms used, the optimizations included to save power and some open issues on the field. They have classified the energy saving techniques in a data center into static and dynamic methods. They have included in the Static Power Management (SPM) class all the optimization methods applied at the design time at the circuit, logic, architecture and system levels. They have included in the Dynamic Power Management (DPM) the techniques, methods and strategies for run-time adaptation of the system behavior. They have also discussed some of the network optimization techniques applied to save energy as the work of McGeer *et al.* (2010) and Fang *et al.* (2013) among others and they have noticed that most of the existing works are evaluated for the flat tree topology. They have proposed that the performance variation over other network topologies needs to be addressed and it is still an open issue. They have observed that the first fit placement algorithm is used for VM placement when network elements are optimized. Ranjana e Raja (2013) have also categorized the process of choosing the right physical machine to host a virtual machine (or VM placement) as constraint programming, whose goal is to maximize a global utility function by composing SLA fulfillment and operating costs, stochastic integer programming, which is useful in cases where actual demands are not known but the distribution of demands is known or can be estimated, and bin packing algorithms, which can be used to find the actual mapping of virtual machines to physical machines. While they focus on VMs consolidation, the work in Ranjana e Raja (2013) assumes that minimizing the number of hosts is the best strategy. On this dissertation, some heuristics that consume less power on non-linear power models using even more hosts are presented.

Verma *et al.* (2008) have investigated the problem of dynamic placement of applications in virtualized systems, while minimizing the power consumption and maintaining the SLA. The authors have proposed the pMapper placement framework. They have implemented the pMapper architecture with the first fit decreasing bin packing algorithm. The results have showed that the approach saves about 25% of power. The major drawback of this work is that it only uses the bin packing approach for the VM placement. While they focus on VMs consolidation, they consider a specific HPC cluster and not an IaaS cloud. They consider CPU and RAM memory and the power model based on the hardware they have used, nevertheless the authors do not mention how to run the algorithms on other power models.

Feller *et al.* (2011) have presented an Ant Colony (ACO) power aware optimization technique for VM placement. They claim to be the first of its kind to apply ACO in dynamic workload placement to conserve energy. However, there is a high cost of computation time due to searching for optimum placement. The impact of co-migration on the placement algorithm is not discussed by the authors. Although their focus is to reduce the number of hosts to serve the VMs, they maximize the number of VMs per host and therefore they do not consider non-linear power models. Furthermore they do not provide details on the power models used.

Authors in Wei *et al.* (2011) propose an algorithm that calculates a fitness value for a VM, Physical Machine (PM) pair based on available memory, network and disk resource of the PM and the demand of migrating VM for these resources. The VMs are arranged in a decreasing order of their fitness value for a given PM. If the fitness value is less than zero then the PM cannot accommodate the VM on migration and the next PM is checked. They have compared their algorithm with a random placement. We have done something similar, nevertheless we do not order the VMs in a decreasing order, instead we choose the best fitting VM. To do so we have applied a modification of the knapsack problem. We also propose a binary coded genetic algorithm, in this case we reinforce our fitness function taking into consideration other resources like CPU, RAM memory and I/O.

Wu *et al.* (2012) propose a different optimizing strategy based on Simulated Annealing theory. They have compared their algorithm and it provided a better performance when compared to the First Fit Decreasing (FFD) algorithm by 25%. However, the problem observed is the time spent by the computation during the algorithm's execution to get an optimum result when the number of the VMs and hosts increase. The proposed solution works well with static VM placement but may not perform well with dynamic VM consolidation due to frequent reshuffling of VMs. Their work differs from ours because they use a linear power model, while our algorithms can use both a linear power model or a SPECpower benchmark power profile.

Al Shayeji e Samrajesh (2012) propose an algorithm where the least loaded host is selected to be switched to a lower power mode. After the server selection, all the VMs are migrated to other hosts. A power off threshold value is calculated for each physical server. To avoid many migrations a counter is maintained. The servers that are below a threshold and with the lowest counters are selected as servers to be freed. The algorithm is evaluated for static and dynamic VM workloads. The performance is compared to a dynamic round robin and a random choice migration algorithm. The performance of the algorithm depends on the optimal choice of the threshold values, however they have not discussed the details on how to calculate these values. While they consider CPU, RAM memory and I/O, they do not specify which power model they use.

Most of the studies focus only on CPU observation. This makes sense since most of the Quality of Service (QoS) and availability are in terms of CPU. However with the growth of big data phenomena and the proliferation of Solid State Disks (SSD) a tremendous performance increase in terms of I/O can been observed. In a similar way the advancements on the fields of networking and fiber connections are delivering fast fiber network connections to the households. For these reasons networking and I/O QoS parameters will probably start to increasingly gain momentum in the near future. There are many research opportunities on the field since there can be many criteria to be explored.

We have proposed several algorithms where the network resource was considered. However, due to the lack of real IaaS VMs network traces we have decided not to include the networking resource in this work, instead we have added new features to the framework to allow the use of the traces from the Google Cluster Data project. These traces include data from CPU, RAM memory and disk usage.

Furthermore, the available cloud simulators nowadays consider CPU as the decision factor, so developing a simulation toolkit that allows to be easily extended with other metrics will fulfill a gap in the state of the art. CloudSim Calheiros *et al.* (2011) is probably the most popular simulation tool for cloud computing environments. It is developed at the University of Melbourne, Australia. CloudSim is made in Java on top of GridSim. Due to the lack of some important features, new cloud simulators based on CloudSim have emerged like WorkflowSim, CloudAnalyst. Despite of the important features of CloudSim and being an event driven simulator, it only supports a single resource: CPU.

CloudAnalyst CloudAnalyst is a cloud simulation tool developed at the University of Melbourne, Australia. It is developed in Java and based on CloudSim. CloudAnalyst focuses on the evaluation of performance and cost of large-scale geographically distributed cloud systems having large user workloads.

GreenCloud GreenCloud is a cloud simulation tool developed at the University of Luxembourg. It is built on top of NS-2. It focuses on energy-aware environments by considering energy on network equipment such as computing servers, communication links, network switches, etc. GreenCloud is made in C++. Since it has been developed on top of NS-2, it focus on monitoring and optimization of communication protocols and network infrastructure.

iCanCloud iCanCloud is a cloud simulation tool developed at the University Carlos III in Madrid, Spain. iCanCloud allows to customize the hypervisor class and it includes an Amazon cloud model. It is developed on C++ and it focuses on predicting the trade-offs between cost and performance of a given set of applications to be executed.

Chapter 4

Energy Efficient Virtual Machines Consolidation Algorithms

There is a relationship between the total power consumption by a server and its CPU utilization as observed in Fan *et al.* (2007). Basically their model proposes that power consumption by a server grows linearly with the growth of the CPU utilization.

CPU utilization based models are able to provide an accurate prediction for CPU-intensive applications, however they tend to be inaccurate for other types of prediction like Network, I/O and memory intensive applications, as concluded in Dhiman *et al.* (2010). For this reason, it is important to include other resource usage indicators in the power consumption models to try to increase the accuracy of the predictions.

This section presents the three proposed algorithms to allocate VMs considering the energy consumption of physical machines modeled with network, CPU, I/O, RAM and virtual memory usage. The first algorithm consolidates the VMs based in the knapsack problem. The second algorithm consolidates the VMs using Evolutionary Computation (EC). The third algorithm consolidates the VMs based in the knapsack problem while considering to double the virtual memory capacity.

We can define the problem of energy efficient virtual machines consolidation as follows. Given sets of:

- Physical hosts $H = \{H_1, \ldots, H_n\}$ of size n
- Available resources $\vec{C}_p = (C_{p,1}, \ldots, C_{p,d})$ of size d
- Virtual Machines $V = \{V_1, \ldots, V_m\}$ of size m
- Used resources $R = \{R_1, \ldots, R_d\}$ of size d
- VM$_j$ resource demands $\vec{r}_j = (r_{j,1}, \ldots, r_{j,d})$ of size d

Each physical host H_i has a d-dimensional available resource capacity as a vector $\vec{C}_i = (C_{i,1}, \ldots C_{i,d})$, for example, Network, CPU, RAM, I/O, etc. All the virtual machines resources usage $vm_j \in V$ are represented by a d-dimensional resource demand vector $\vec{r}_j = (r_{j,1}, \ldots, r_{j,d}) \in [0, 100]^d$. These values represent a snapshot of resources usage, in percentage, by the VM$_j$.

The objective is to minimize the physical hosts used, which can also be seen as maximize the number of VMs per host constrained to the stochastic resources usage.

4.1 Knapsack Problem based Heuristic

In this first approach the VMs placement problem has been modeled as a Multidimensional Bin Packing (MDBP) approach. In this approach the following decision variables need to be found: The physical host allocation variable h_p and the virtual machine allocation variable $x_{i,p}$. h_p is 1 if the physical host p is used by some VM or 0, otherwise. $x_{i,p}$ is 1 if the virtual machine i is assigned to the host p or 0, otherwise.

The goal is to minimize the number of physical hosts:

$$\text{Minimize} \sum_{p=1}^{n} h_p \tag{4.1}$$

Subject to the following constraints:

$$\sum_{i=1}^{m} r_{i,k} \times x_{i,p} \leq C_{p,k} \times h_p, \forall p \in \{1, \ldots, n\}, \forall k \in R \tag{4.2}$$

$$\sum_{i=1}^{n} x_{i,p} = 1, \forall i \in \{1, \ldots, m\} \tag{4.3}$$

Where constraint (4.2) ensures that the capacity of each physical host is not exceeded and constraint (4.3) guarantees that each virtual machine is assigned to exactly one host.

MDBP is a variation of the Multiple-Choice Multidimensional Knapsack Problem (MMKP) which is an NP-Hard problem. The first proposed algorithm consists in solving the standard knapsack problem n-times (one per each host) using a list of VMs request. The name given to this approach is: *Iterated-KSP*.

The Algorithm 1 shows the function that implements the Iterated-KSP. The algorithm first prepares a list of constraints (line 2) for each resource. The summation of each resource used by the VMs within a host must be less than 99%. The constraints also include assigning a weight on each VM which will be the criteria to be maximized by the algorithm. In the simulation we used equal weights for all the VMs. Nevertheless, according to different strategies, this could be changed. Then, a list of hosts (line 3) and unplaced VMs (line 4) are created. A loop iterates over the available physical hosts to start placing the VMs (line 5). An instance of the KSP problem is defined (line 6) and solved (line 7) producing the VMs placement on that particular host. Finally the VMs are placed (line 8) and removed from the unplaced VMs list (line 9). The resulting placement is returned for that host (line 11).

The Iterated-KSP algorithm shown could be improved for specific needs, for instance, considering the use of virtual memory it is possible to modify the algorithm by doubling the RAM capacity. In this case, it is possible to achieve a better placement on a data center that run VMs with high RAM demand. This idea of considering the existence of virtual memory (double of the current physical memory) is applied in the third algorithm proposed, which is called *Iterated-KSP-Mem*. Algorithm 1 is also executed to find the allocation in this algorithm. The only difference to the Iterated-KSP is that the amount of available memory considered is the double of the physical memory.

Algorithm 1 Iterated-KSP Strategy

```
1: function ITERATED-KSP(...)
2:     constraints ← for every $d$-dimension to be $\leq 99$
3:     create a list of hosts
4:     create an unplaced VMs list
5:     for each host do
6:         problem ← KSP(unplaced VMs list, constraints)
7:         vms ← problem.solve()
8:         place the VMs on the host
9:         remove the placed VMs from unplaced VMs list
10:    end for
11:    return placement
12: end function
```

4.2 Evolutionary Computation based Heuristic

The Evolutionary Computation model separates problem-specific computation from algorithm-specific computation. The algorithm-specific computation is related to bio-inspired algorithms while the problem-specific computation is related on how solutions to the problem look like and how such solutions are evaluated Garrett (2013). The name given to this algorithm is: *Iterated-EC*.

The Iterated-EC algorithm is presented on Algorithm 2. The algorithm starts by defining the population and tournament size (lines: 2 and 3) and the number of evaluations (line 4). Then, a list of hosts (line 5) and unplaced VMs (line 6) are created. A loop iterates over the available physical hosts to start placing the VMs. All of these lines so far are problem-specific components. There are algorithm-specific components that can be found on any evolutionary computation problem (lines: 8 to 20), which includes the terminator ($\mathscr{T}$), observer ($\mathscr{O}$) and variator ($\mathscr{V}$). Finally after a placement solution has been found, the VMs are placed (line 22) and removed from the unplaced VMs list (line 23). The resulting placement is returned for that host (line 26). Similar to the Iterated-KSP algorithm the objective is to maximize the number of VMs per physical host.

The generator function ($\mathscr{G}$) generates possible solutions where each bit corresponds to including/excluding the VM at that host, giving a 1% chance of each bit being one, since is desired that the vast majority of the candidates to be zeroes.

The evaluation function ($\mathscr{E}$) is used on every generation evaluation within the Algorithm 2 (line 16). This function first calculates the totals of the sums per resource of the VMs from the candidate solution, then it calculates the maximum between zero and the per resource sum. A fitness calculation is performed as the difference of number of VMs within the proposed candidate solution and the sum of the constraints. The only scenario where fitness is positive is when the sum of constraints is exactly zero (proposed VMs fit within the host). The generation which produces the higher fitness value will be proposed as the VM placement solution for that currently evaluated host.

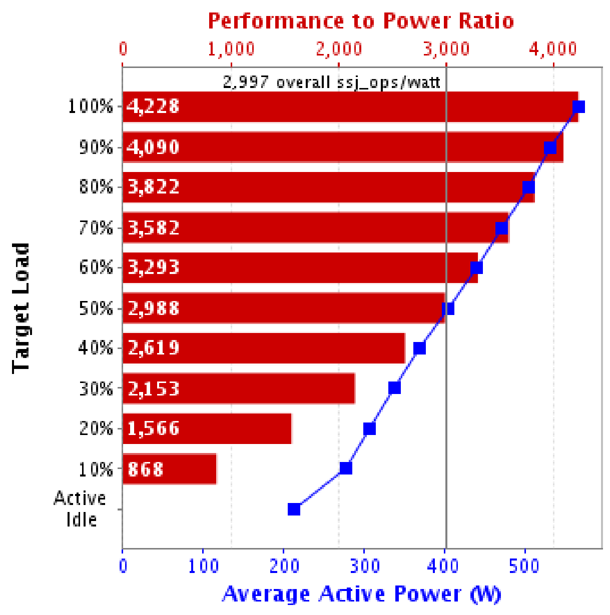
The terminator function ($\mathscr{T}$) defines the termination criteria used, in general when a number of generations has been achieved. The observer function ($\mathscr{O}$) is optionally called for monitoring on each generation. The variator function ($\mathscr{V}$) defines the mutations to be used as already described.

Algorithm 2 Iterated-EC Strategy

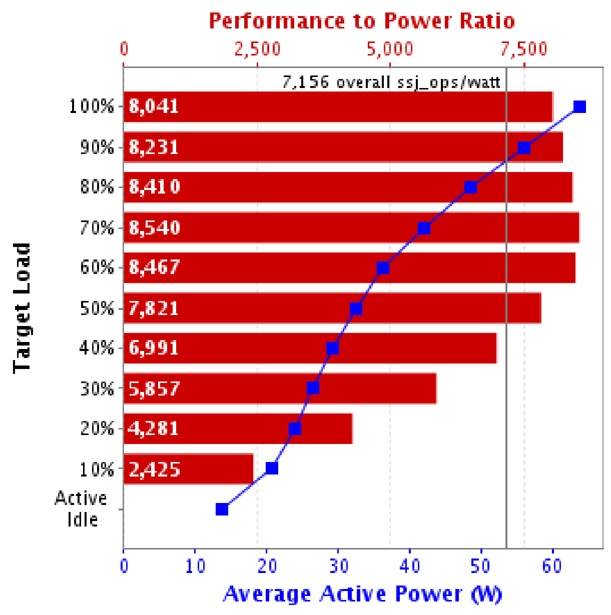
```
1: function ITERATED-EC(...)
2:     population size ← 50
3:     tournament size ← 25
4:     number of evaluations ← 2500
5:     create a list of hosts
6:     create an unplaced VMs list
7:     for each host do
8:         Create the initial population using $\mathscr{G}$
9:         Evaluate the initial population using $\mathscr{E}$
10:        Set the number of generations to 0
11:        while the terminator $\mathscr{T}$ is not true Loop do
12:            Choose parents via the selector $\mathscr{S}$
13:            Initialize offspring to the parents
14:            for each variator $\mathscr{V}$ loop do
15:                Set offspring
16:                Evaluate offspring using the evaluator $\mathscr{E}$
17:                Update the number of evaluations
18:                Replace, Migrate and Archive population
19:                Increment the number of generations
20:                Call the observer $\mathscr{O}$ on the population
21:            end for
22:            place the VMs on the host
23:            remove the placed VMs from unplaced list
24:        end while
25:    end for
26:    return placement
27: end function
```

4.3 Linear and Non Linear SPECpower Models Heuristics

The algorithms proposed on the previous subsections focus on maximizing the number of VMs per hosts considering several aspects like CPU, RAM memory and disk. However, modern servers are equipped with large amounts of memory, due to hardware evolution on multi-core CPUs and the proliferation of virtualization. This fact has made the memory to begin dominating the power consumption on some server models. This might lead some servers to have a non-linear power consumption behavior.



(a) *Sugon I840-G25 (linear)*
SPECPower Sugon I840-G25 Server



(b) *Fujitsu PRIMERGY RX1330 M1 (non-linear)*
SPECPower Fujitsu PRIMERGY RX1330 Server

Figure 4.1: *Linear and non-linear SPECpower models*

Figure 4.1 shows a linear and a non-linear SPECpower models used on this work. In general, the SPECpower benchmark uses a metric which is the Server Side Java Operations (ssj_ops) in relation to the consumed watts. The Figure 4.1a is considered a linear model because the power consumption grows as the ssj_ops/watt ratio grows. Therefore the optimum ratio is 100%. Lets illustrate this with an example of a single resource comparison by observing the blue line (not to be confused with the bars) of Figure 4.1a. Lets imagine two placements, the first placement (A) places all the VMs in two servers, the first server has a 90% CPU load consuming 531 Watts and the second server has a 10% CPU load consuming 279 Watts. This placement would have an overall power consumption of 810 Watts. Now lets imagine a second placement (B), that places all the VMs into a single server having a 100% CPU load consuming 567 Watts. On a linear model, the second (B) placement would be more power efficient than the first placement (A) due to the fact that it consumes less power.

This is not the same behavior observed on the Figure 4.1b where there is an optimum ssj_ops/watt ratio at 70% instead. Lets illustrate this also with an example of a single resource comparison. Lets imagine two placements, the first placement (A) places all the VMs in two servers, the first server having 60% CPU load consuming 36.2 Watts instead of targeting its load to the maximum CPU capacity, and the second server with 50% CPU load consuming 32.6 Watts. This placement would have an overall power consumption of 68.8 Watts. Now lets imagine a second placement (B), that places all the VMs into the same two server as follows, having the first server with a 100% CPU load consuming 63.7 Watts and the second server with a 10% CPU load consuming 20.8 Watts. This placement would have an overall power consumption of 84.5 Watts. On a non-linear model the first (A) placement would be more power efficient than the second placement (B) due to the fact that it consumes less power.

By modifying the previous algorithms to allow a variable maximum target load, it is possible to select the optimum server ssj_ops/watt ratio. By repeating this process across each server on the data center it is possible to produce a better VMs placement than with the linear heuristics. While a non-linear placement is better on a non-linear model there is a local vs. global optimum dilemma, this means that, it would be possible to have variations on the optimum ssj_ops/watt ratio that would produce a global optimum placement even when all the servers do not use the best ssj_ops/watt ratio. Therefore even when greedy approaches produce better placements than a linear heuristic, they do not necessarily produce a global minimum power aware placement. This idea is explained with more details on Section 5.2.2.

Chapter 5

Performance Evaluation

This chapter presents an overview of the pyCloudSim simulation framework on Section 5.1. The linear and non-linear models are presented on Section 5.2. The experiments using the PlanetLab traces are presented on Section 5.3, while the experiments using the Google traces are presented on Section 5.4.

5.1 The pyCloudSim Simulation Framework and Methodology

The pyCloudSim simulation framework [1] has been developed to study power-efficient VMs machine allocation. pyCloudSim has been developed in Python, which is a high level programming language widely used for scientific purposes. Some libraries have been used from the *Python Package Index* (pypi). Among them the most important ones are:

- *OpenOpt* Kroshko (2013), which is a framework for numerical optimization developed by the Optimization department of the Cybernetics Institute and sponsored by the National Academy of Sciences of Ukraine. OpenOpt was originally developed in Matlab and afterwards ported to Python using *NumPy* for numerical work. OpenOpt supports about 30 different solvers for optimization problems both free and commercial (like CPlex, Knitro and Matlab), focusing on open source solvers. Some solvers are written in C or Fortran.

- *FuncDesigner* FuncDesigner , which is a Computer Algebra System (CAS) used from within OpenOpt.

- *Inspyred* Garrett (2013) is a library of biologically-inspired algorithms including evolutionary computation, swarm intelligence, and neural networks. The Inspyred library grew out of insights from Jong e Jong (2002). The goal of the library is to separate the problem-specific computation from algorithm-specific computation in a clean way so as to make algorithms as general as possible across a range of different problems. Any bio-inspired algorithm has at least two aspects that are entirely problem-specific: how solutions to the problem look like and how such solutions are evaluated Garrett (2013).

- *NumPy* NumPy , which is a mathematical library written in Python, C and Fortran that allows to perform powerful operations on n-dimensional array objects. It also include wrappers

[1] http://pycloudsim.albertdelafuente.com, Last access on March 15, 2015

around most Fortran functions included in the BLAS BLAS (Basic Linear Algebra Subprograms) and LAPACK LAPACK (Linear Algebra PACKage) libraries.

- *Matplotlib* Matplotlib which is a plotting library for Python and Numpy. It allows to do line plots, histograms, scatter plots, 3D plots and many others.

Graphically, pyCloudSim can be structured in the block diagram showed on Figure 5.1. The most important parts are the *Global Manager* and the *Simulation Manager*. The simulation manager is in charge of executing the strategies given the simulation parameters. The global manager also communicates with the *Physical Machine Manager* and *Virtual Machine Manager*. These modules are responsible for managing the physical hosts and virtual machines respectively.

The virtual machine manager is also responsible for mapping the VMs with traces either from the PlanetLab project Planetlab Traces or from the Google Cluster workload data Google Cluster Data Traces . This is performed on the initialization time, using the *Trace Manager*.

The *Physical Machine Manager* includes the power estimation routines which requires the SPECpower data to estimate power consumption based on real server measurement as already introduced on Section 2.5.1.

After the experiments are finished, the data is summarized to calculate the best, worst and average cases and the confidence intervals. Finally the data is represented graphically on different type of plots representing several aspects of the placement.

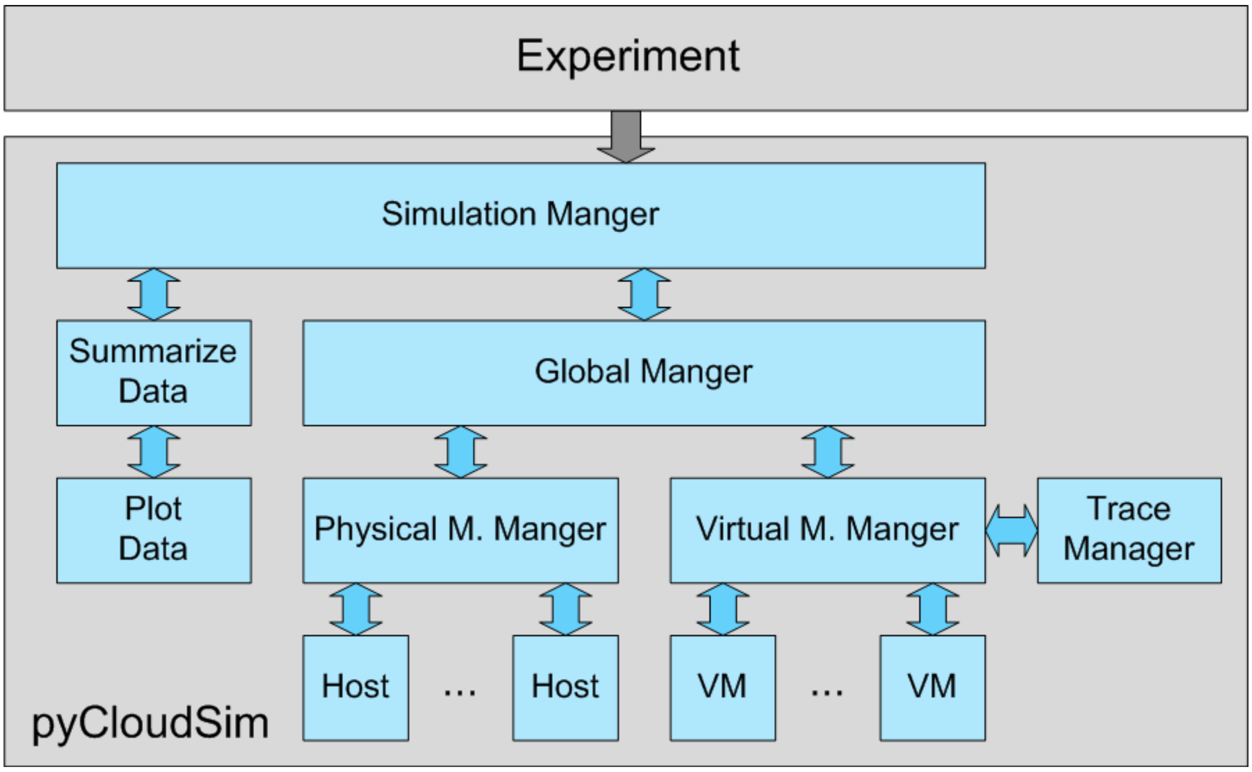


Figure 5.1: *pyCloudSim block diagram*

The simulation framework is based on the Algorithm 3 that iterates over the available (unplaced) physical hosts scenarios $\langle PMS \rangle$ (line 2) and later over the VMs scenarios $\langle VMS \rangle$ (line 3) to determine a placement using a given strategy $\mathcal{S}$ for that scenario of physical hosts and VMs. The strategies can be Energy Unaware, Iterated-KSP, Iterated-KSP-Mem and Iterated-EC.

We employed a general heuristic architecture to perform the placement of the VMs on the hosts. The heuristic works as follows: the placement will be performed by sequentially evaluating the next suitable physical host that can handle the workload of a given VM, if any.

On the strategies that are energy aware, the hosts that are underloaded (without VMs workload) are automatically suspended by an energy aware strategy with the consequent energy savings. Later, they can be reactivated when required using mechanisms like Wake-on-Lan Zelkowitz (2011), which allows a computer to be turned on.

To evaluate the gains of the strategies, they are compared with an energy unaware First Fit (FF) algorithm, called Energy Unaware strategy. The aim was to evaluate if a single solution like a FF would be better. The strategies were not compared against other existing on the literature because, for the best of our knowledge, there is no solution that considers multiple resources and a power-efficient strategy simultaneously. The Energy-Unaware strategy evaluates the hosts sequentially and chooses the next VM on the queue that can be allocated on that host. If the host has enough resources to allocate that VM, then it is placed within that host, otherwise that VM is left on the queue for later processing and another VM is selected from the queue until no more VMs or hosts are left.

Algorithm 3 Virtual Machines placement framework

```
Input: A Strategy: $\mathcal{S}$
Input: List of physical hosts scenarios: $\langle PMS \rangle$
Input: List of virtual machines scenarios: $\langle VMS \rangle$
Output: A placement solution

1: function SIMULATE-STRATEGY(...)
2:     for $\mathscr{P} \in \langle PMS \rangle$ do
3:         for $\mathscr{V} \in \langle VMS \rangle$ do
4:             Solution-scenario ← Simulate $\mathcal{S}$ with $\mathscr{P}$, $\mathscr{V}$
5:         end for
6:     end for
7: end function
```

The consumed power estimation is performed using two different models. The first one implemented was a linear model based on the CPU utilization, as proposed in Fan *et al.* (2007) without loss of generality since the focus is to efficiently place the VMs regarding their resources use. The second power model used is based on a SPECpower system profile benchmark. pyCloudSim allows to specify a different SPECpower profile for each host on the cloud and the power consumption is based on the CPU utilization.

It is important to conduct experiments using workload traces from real systems rather than artificially generated ones. So, two approaches were used to define the workloads.

The first approach was to use workloads from the Planetlab project Planetlab Traces , therefore we analyzed more than 11,776 24-hour long traces provided as part of the CoMon project CoMon - A Monitoring Infrastructure for Planetlab , a monitoring infrastructure for PlanetLab. 10 days of workload traces collected during March and April 2011 have been randomly chosen as the data set. The traces include data on the CPU utilization collected every 5 minutes from more than a thousand VMs deployed on servers located in more than 500 places around the world.

The second approach was to use workloads from the Google Cluster data Google Cluster Data Traces repository [2]. More than 40 Gigabytes of compressed text data have been analyzed and some of the workloads were selected to be used as the VMs.

5.2 Linear and Non-linear SPECpower Benchmark Profiles

The experiments have been performed using real SPECpower benchmark profiles. We used the only two available server profiles available during the third quarter of 2014. The two next subsections presents the profiles.

5.2.1 Linear SPECpower Benchmark Profile

The first SPECpower profile used is from a Sugon I840-G25 server with 4 chips, 48 cores and 96 total threads. This server has 128GB of RAM memory, an average of 213 Watts of power consumption at idle state, and an average of 567 Watts of power consumption at full-load (100%). The server power consumption profile is shown on Figure 5.2

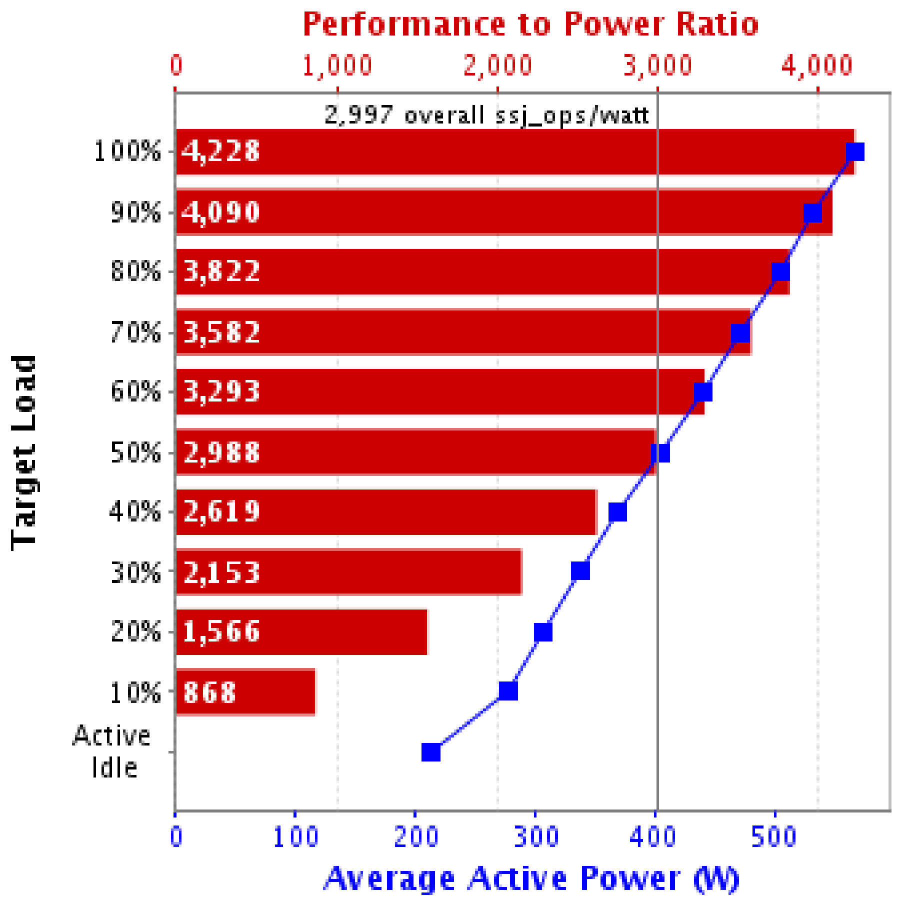


Figure 5.2: *SPECpower with linear characteristics of a Sugon I840-G25 Server SPECPower Sugon I840-G25 Server*

While there is not a strict linear relationship between the target loads, there is a discrete linear growing relationship. In other words, each target load consumption could be expressed as a constant

[2]https://code.google.com/p/googleclusterdata/, Last access on March 15, 2015

multiplied by the prior power consumption value, nevertheless those constants are not the same.

We will consider a linear model, any model that when the target load increases, the power ssj_ops/watt ratio also increases even if it is not proportionally, as shown on Figure 5.2. In that sense, this model will be referred as an example of a SPECpower model with a linear behavior.

Lets first analyze a simple example of 11 VM requests to be placed into 7 hosts. In this case it will be assumed that all the hosts have a linear model power consumption. The resources utilization of the VMs can be found on Table 5.1, please note that no virtualization overhead is assumed therefore, the usage of these resources are based only on the workload requirements.

VM-id	CPU *(%)*	RAM *(%)*	I/O *(%)*
VM001	30.00	8.00	0.30
VM002	30.00	8.00	0.26
VM003	20.00	8.00	1.31
VM004	20.00	5.00	1.63
VM005	10.00	5.00	2.12
VM006	30.00	8.00	1.31
VM007	30.00	10.00	0.30
VM008	20.00	10.00	1.63
VM009	10.00	10.00	0.26
VM010	10.00	5.00	2.12
VM011	10.00	5.00	1.31

Table 5.1: *Simple example of 11 VMs workload scenario*

After applying the Energy Unaware heuristic, the proposed allocation will result on Table 5.2. On the placement, the VM007, VM009, VM008 and VM001 have been placed on the first host. The VM003, VM004, VM010, VM011 and VM002 have been placed on the second host. The VM005 and VM006 have been placed on the third host. The CPU, RAM and I/O columns represent the resource percentage to be used by the VMs when placed. The Av. CPU, Av. RAM and Av. I/O represents the available CPU, RAM and I/O resources respectively. The last column represents the incremental power use per host in Watts after a VM is placed on that host, for instance on the first host H001, after placing the VM007, the power consumption is 337 Watts. After placing the VM009 the power consumption rises to 370 Watts and so forth. Note that the hosts H004, H005, H006 and H007 are not hosting any VM nevertheless they do have an idle power consumption, and therefore are shown on the table to illustrate the power consumption of each host. The last row summarizes the power consumption of the seven hosts all together.

Host-id	VM-id	CPU *(%)*	RAM *(%)*	I/O *(%)*	Av. CPU *(%)*	Av. RAM *(%)*	Av. I/O *(%)*	Inc. Power *(W)*
H001	VM007	30.00	10.00	0.30	70.00	90.00	99.70	337.00
	VM009	10.00	10.00	0.26	60.00	80.00	99.44	370.00
	VM008	20.00	10.00	1.63	40.00	70.00	97.81	439.00
	VM001	30.00	8.00	0.30	10.00	62.00	97.51	531.00
H002	VM003	20.00	8.00	1.31	80.00	92.00	98.69	307.00
	VM004	20.00	5.00	1.63	60.00	87.00	97.06	370.00
	VM010	10.00	5.00	2.12	50.00	82.00	94.94	404.00
	VM011	10.00	5.00	1.31	40.00	77.00	93.63	439.00
	VM002	30.00	8.00	0.26	10.00	69.00	93.37	531.00
H003	VM005	10.00	5.00	2.12	90.00	95.00	97.88	279.00
	VM006	30.00	8.00	1.31	60.00	87.00	96.57	370.00
H004	-	0.00	0.00	0.00	100.00	100.00	100.00	213.00
H005	-	0.00	0.00	0.00	100.00	100.00	100.00	213.00
H006	-	0.00	0.00	0.00	100.00	100.00	100.00	213.00
H007	-	0.00	0.00	0.00	100.00	100.00	100.00	213.00
								2,284.00

Table 5.2: *Simple example of 11 VMs placement using the Energy Unaware algorithm*

Analogously, the same experiment was repeated on the same scenario using the Iterated-KSP algorithm. The returned placement is shown on Table 5.3. Note that the hosts 4, 5, 6 and 7 have been suspended to a lower state of power consumption by the algorithm and therefore they do not appear on the table. The allocation returned by the Iterated-KSP lead to a lower overall power consumption when compared to the allocation returned by the Energy Unaware (1,435 Watts versus 2,284 Watts)

Host-id	VM-id	CPU *(%)*	RAM *(%)*	I/O *(%)*	Av. CPU *(%)*	Av. RAM *(%)*	Av. I/O *(%)*	Inc. Power *(W)*
H001	VM003	20.00	8.00	1.31	80.00	92.00	98.69	307.00
	VM004	20.00	5.00	1.63	60.00	87.00	97.06	370.00
	VM005	10.00	5.00	2.12	50.00	82.00	94.94	404.00
	VM008	20.00	10.00	1.63	30.00	72.00	93.31	471.00
	VM009	10.00	10.00	0.26	20.00	62.00	93.05	505.00
	VM010	10.00	5.00	2.12	10.00	57.00	90.93	531.00
	VM011	10.00	5.00	1.31	0.00	52.00	89.62	567.00
H002	VM001	30.00	8.00	0.30	70.00	92.00	99.70	337.00
	VM006	30.00	8.00	1.31	40.00	84.00	98.39	439.00
	VM007	30.00	10.00	0.30	10.00	74.00	98.09	531.00
H003	VM002	30.00	8.00	0.26	70.00	92.00	99.74	337.00
								1,435.00

Table 5.3: *Simple example of 11 VMs placement using the Iterated-KSP algorithm*

Likewise using the Iterated-EC algorithm as shown Table 5.4. Note that analogously to the Iterated-KSP algorithm, the hosts 4, 5, 6 and 7 have been suspended to a lower state of power consumption by the algorithm and therefore they do not appear on the table. Also note that the host H001 placement returned by both algorithms Iterated-KSP and Iterated-EC is the same, the H002 and H003 placement differs.

Host-id	VM-id	CPU *(%)*	RAM *(%)*	I/O *(%)*	Av. CPU *(%)*	Av. RAM *(%)*	Av. I/O *(%)*	Inc. Power *(W)*
H001	VM003	20.00	8.00	1.31	80.00	92.00	98.69	307.00
	VM004	20.00	5.00	1.63	60.00	87.00	97.06	370.00
	VM005	10.00	5.00	2.12	50.00	82.00	94.94	404.00
	VM008	20.00	10.00	1.63	30.00	72.00	93.31	471.00
	VM009	10.00	10.00	0.26	20.00	62.00	93.05	505.00
	VM010	10.00	5.00	2.12	10.00	57.00	90.93	531.00
	VM011	10.00	5.00	1.31	0.00	52.00	89.62	567.00
H002	VM001	30.00	8.00	0.30	70.00	92.00	99.70	337.00
	VM002	30.00	8.00	0.26	40.00	84.00	99.44	439.00
	VM006	30.00	8.00	1.31	10.00	76.00	98.13	531.00
H003	VM007	30.00	10.00	0.30	70.00	90.00	99.70	337.00
								1,435.00

Table 5.4: *Simple example of 11 VMs placement using the Iterated-EC algorithm*

The experiments with the linear model can be found on Section 5.3.2 for the PlanetLab traces and on Section 5.4.2 for the Google Cluster traces.

5.2.2 Non-Linear SPECpower Benchmark Profile

The second SPECpower profile used is from a Fujitsu PRIMERGY RX1330 M1 server with 1 chip, 4 cores and 8 total threads. This server has 16GB of RAM memory, an average of 13.8 Watts of power consumption at idle state, and 63.7 Watts of power consumption at full load (100%). The server power consumption profile is shown on Figure 5.3

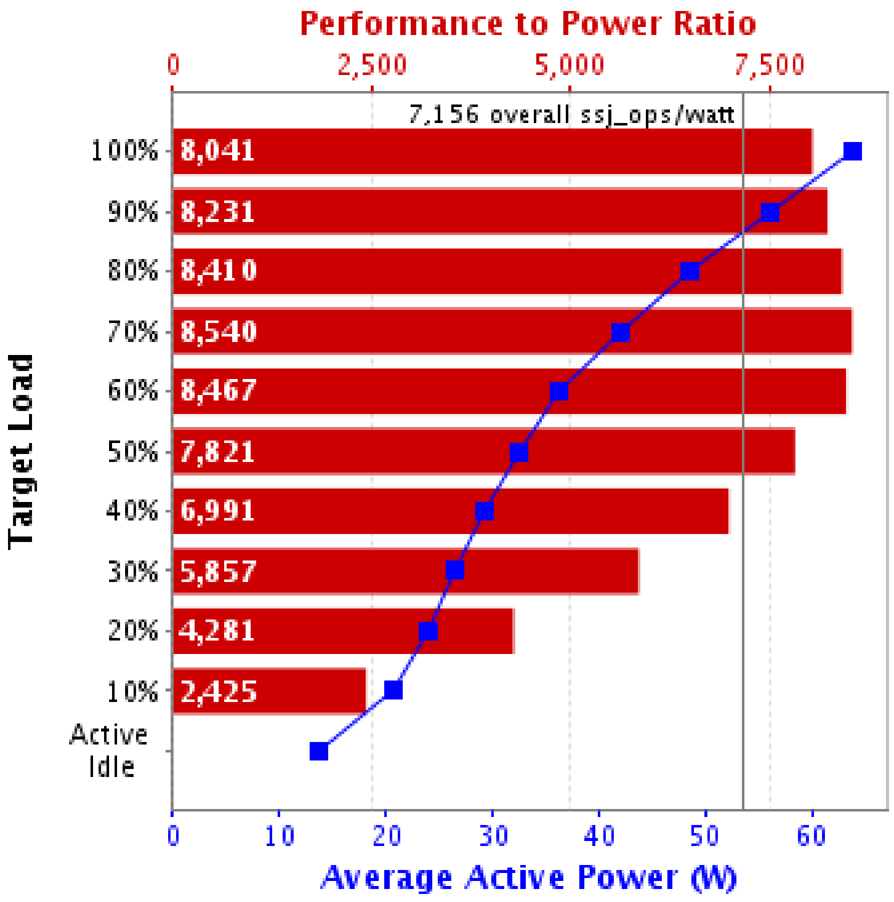


Figure 5.3: *SPECpower with non-linear characteristics of a Fujitsu PRIMERGY RX1330 M1 Server SPECPower Fujitsu PRIMERGY RX1330 Server*

We can observe that when the target load increases, the performance does not increases linearly, in fact, it is better to have a 70% target load on this sever since it will have its optimal ssj_ops/watt ratio. Therefore this server has a non-linear model power consumption.

Note that the server with a linear model power consumption presented on Section 5.2.1 has more hardware capacity (number of chips, cores and RAM), therefore both servers might have different usage scopes within a data center depending on how the applications are distributed. For instance the servers with a linear power consumption model could be dedicated to higher RAM demanding tasks.

It is important to consider which optimization technique to use with non-linear models since using a linear optimization (for example, maximizing the number of VMs) could lead to higher power consumption rates. Lets show an example of this by analyzing the allocation of the same example introduced on Section 5.2.1. In this case it is assumed that all the hosts have a non-linear model power consumption. The resources utilization of the VMs can be found on Table 5.1, please note that no virtualization overhead is assumed therefore, the usage of this resources are based only on the workload requirements.

Host-id	VM-id	CPU *(%)*	RAM *(%)*	I/O *(%)*	Av. CPU *(%)*	Av. RAM *(%)*	Av. I/O *(%)*	Inc. Power *(W)*
H001	VM003	20.00	8.00	1.31	80.00	92.00	98.69	23.90
	VM004	20.00	5.00	1.63	60.00	87.00	97.06	29.10
	VM005	10.00	5.00	2.12	50.00	82.00	94.94	32.60
	VM008	20.00	10.00	1.63	30.00	72.00	93.31	42.00
	VM009	10.00	10.00	0.26	20.00	62.00	93.05	48.60
	VM010	10.00	5.00	2.12	10.00	57.00	90.93	55.90
	VM011	10.00	5.00	1.31	0.00	52.00	89.62	63.70
H002	VM001	30.00	8.00	0.30	70.00	92.00	99.70	26.30
	VM006	30.00	8.00	1.31	40.00	84.00	98.39	36.20
	VM007	30.00	10.00	0.30	10.00	74.00	98.09	55.90
H003	VM002	30.00	8.00	0.26	70.00	92.00	99.74	26.30
								145.90

Table 5.5: *Simple example of 11 VMs placement using the Iterated-KSP algorithm with a linear heuristic*

The Table 5.5 shows the placement resulted of applying the Iterated-KSP algorithm with a linear heuristic aiming to minimize the number of hosts (equivalent to maximize the number of VMs per host). The CPU, RAM and I/O columns show the resources required per VM. The Av. CPU, Av. RAM and Av. I/O shows the available resources on that host, therefore it decreases after placing new VMs on subsequent rows. The Inc. Power column represents the incremental power consumed after placing a VM. Note that this placement requires 3 hosts and consumes 145.90 Watts.

Host-id	VM-id	CPU *(%)*	RAM *(%)*	I/O *(%)*	Av. CPU *(%)*	Av. RAM *(%)*	Av. I/O *(%)*	Inc. Power *(W)*
H001	VM003	20.00	8.00	1.31	80.00	92.00	98.69	23.90
	VM005	10.00	5.00	2.12	70.00	87.00	96.57	26.30
	VM008	20.00	10.00	1.63	50.00	77.00	94.94	32.60
	VM009	10.00	10.00	0.26	40.00	67.00	94.68	36.20
	VM010	10.00	5.00	2.12	30.00	62.00	92.56	42.00
H002	VM002	30.00	8.00	0.26	70.00	92.00	99.74	26.30
	VM004	20.00	5.00	1.63	50.00	87.00	98.11	32.60
	VM011	10.00	5.00	1.31	40.00	82.00	96.80	36.20
H003	VM006	30.00	8.00	1.31	70.00	92.00	98.69	26.30
	VM007	30.00	10.00	0.30	40.00	82.00	98.39	36.20
H004	VM001	30.00	8.00	0.30	70.00	92.00	99.70	26.30
								140.70

Table 5.6: *Simple example of 11 VMs placement using the Iterated-KSP algorithm with a non-linear heuristic using the first local optimum (70% target load)*

On Table 5.6 is shown the placement resulted of applying the Iterated-KSP algorithm with a non-linear heuristic using the first optimum of the Fujitsu PRIMERGY RX1330 M1 Server, which is targeting 70% of the load. Note that this placement requires 4 hosts and consumes 140.70 Watts, which demonstrates to be a better placement than the one summarized on Table 5.5, since it consumes less power even when it uses more hosts.

Host-id	VM-id	CPU *(%)*	RAM *(%)*	I/O *(%)*	Av. CPU *(%)*	Av. RAM *(%)*	Av. I/O *(%)*	Inc. Power *(W)*
H001	VM004	20.00	5.00	1.63	80.00	95.00	98.37	23.90
	VM005	10.00	5.00	2.12	70.00	90.00	96.25	26.30
	VM009	10.00	10.00	0.26	60.00	80.00	95.99	29.10
	VM010	10.00	5.00	2.12	50.00	75.00	93.87	32.60
	VM011	10.00	5.00	1.31	40.00	70.00	92.56	36.20
H002	VM007	30.00	10.00	0.30	70.00	90.00	99.70	26.30
	VM008	20.00	10.00	1.63	50.00	80.00	98.07	32.60
H003	VM002	30.00	8.00	0.26	70.00	92.00	99.74	26.30
	VM003	20.00	8.00	1.31	50.00	84.00	98.43	32.60
H004	VM001	30.00	8.00	0.30	70.00	92.00	99.70	26.30
	VM006	30.00	8.00	1.31	40.00	84.00	98.39	36.20
								137.60

Table 5.7: *Simple example of 11 VMs placement using the Iterated-KSP algorithm with a non-linear heuristic using the second local optimum (60% target load))*

Table 5.7 shows the placement after applying the Iterated-KSP algorithm with a non-linear heuristic using the second optimum of the Fujitsu PRIMERGY RX1330 M1 Server, which is targeting 60% of the load. Note that this placement also requires 4 hosts and consumes 137.60 Watts. Therefore this placement is a better placement than the one summarized on Table 5.6, since it consumes less power.

Host-id	VM-id	CPU *(%)*	RAM *(%)*	I/O *(%)*	Av. CPU *(%)*	Av. RAM *(%)*	Av. I/O *(%)*	Inc. Power *(W)*
H001	VM003	20.00	8.00	1.31	80.00	92.00	98.69	23.90
	VM005	10.00	5.00	2.12	70.00	87.00	96.57	26.30
	VM008	20.00	10.00	1.63	50.00	77.00	94.94	32.60
	VM009	10.00	10.00	0.26	40.00	67.00	94.68	36.20
	VM010	10.00	5.00	2.12	30.00	62.00	92.56	42.00
	VM011	10.00	5.00	1.31	20.00	57.00	91.25	48.60
H002	VM004	20.00	5.00	1.63	80.00	95.00	98.37	23.90
	VM006	30.00	8.00	1.31	50.00	87.00	97.06	32.60
	VM007	30.00	10.00	0.30	20.00	77.00	96.76	48.60
H003	VM001	30.00	8.00	0.30	70.00	92.00	99.70	26.30
	VM002	30.00	8.00	0.26	40.00	84.00	99.44	36.20
								133.40

Table 5.8: *Simple example of 11 VMs placement using the Iterated-KSP algorithm with a non-linear heuristic using the third local optimum (80% target load)*

Table 5.8 shows the placement resulted of applying the Iterated-KSP algorithm with a non-linear heuristic using the third optimum state of the Fujitsu PRIMERGY RX1330 M1 Server, which targets 80% of the load. Note that this placement requires 3 hosts and consumes 133.40 Watts, which demonstrates to be the best power-aware placement compared to the placements shown on Tables 5.5, 5.6 and Table 5.7.

Host-id	VM-id	CPU *(%)*	RAM *(%)*	I/O *(%)*	Av. CPU *(%)*	Av. RAM *(%)*	Av. I/O *(%)*	Inc. Power *(W)*
H001	VM003	20.00	8.00	1.31	80.00	92.00	98.69	23.90
	VM004	20.00	5.00	1.63	60.00	87.00	97.06	29.10
	VM005	10.00	5.00	2.12	50.00	82.00	94.94	32.60
	VM008	20.00	10.00	1.63	30.00	72.00	93.31	42.00
	VM009	10.00	10.00	0.26	20.00	62.00	93.05	48.60
	VM010	10.00	5.00	2.12	10.00	57.00	90.93	55.90
H002	VM002	30.00	8.00	0.26	70.00	92.00	99.74	26.30
	VM006	30.00	8.00	1.31	40.00	84.00	98.43	36.20
	VM011	10.00	5.00	1.31	30.00	79.00	97.12	42.00
H003	VM001	30.00	8.00	0.30	70.00	92.00	99.70	26.30
	VM007	30.00	10.00	0.30	40.00	82.00	99.40	36.20
								134.10

Table 5.9: *Simple example of 11 VMs placement using the Iterated-KSP algorithm with a non-linear heuristic using the fourth local optimum (90% of target load)*

We might falsely think that a placement after applying the Iterated-KSP algorithm with a non-linear heuristic using the fourth optimum, which targets 90% of the load would be better. This is not true; as shown on Table 5.9, the placement consumes more than the placement of Table 5.8.

What happens here is a global-local optimum dilemma. Finding the best non-linear placement depend on knowing the global Cloud workload. This is very hard to predict, especially on IaaS Clouds since it heavily depends on the consumer behavior. It is true however that non-linear server models generally perform better with non-linear heuristics.

Note that only the Iterated-KSP has been used on the example since it is the most deterministic of the algorithms.

The non-linear experiments are detailed on Section 5.3.3 use the PlanetLab traces while Section 5.4.3 use the Google Cluster traces.

5.3 Experiments Using Workloads from the PlanetLab Project

The analyzed PlanetLab project workloads were more than 11,776 24-hour long traces provided as part of the CoMon project, a monitoring infrastructure for PlanetLab. The workloads[3] were randomly collected between March and April 2011. The traces include data on the CPU utilization collected every 5 minutes from more than a thousand VMs deployed on servers located in more than 500 places around the world.

In the simulations we considered a scenario of a Cloud IaaS using 100 hosts with VMs allocation requests varying from 16 to 128 VMs with 16 increments. Two types of hosts were considered. One type of server was the Sugon I840-G25 server which has linear power consumption characteristics as presented on Section 5.2.1. The Section 5.3.2 is dedicated to discuss the results of the exper-

[3]https://github.com/vonpupp/planetlab-workload-traces, Last access on March 15, 2015

iments when all the hosts were simulated as being this server. The other type of server was the Fujitsu PRIMERGY RX1330 M1 server which has non-linear power consumption characteristics as presented on Section 5.2.2. The Section 5.3.3 is dedicated to discuss the results of the experiments when all the hosts were simulated as being this server.

Each simulation was repeated 30 times to check if there was a clear tendency, and later the data was reduced to three cases: best, worst and average case (mean). The simulations were executed in a VM with four dedicated cores and 10 Gigabytes of RAM hosted by a physical machine with an octo-core Intel(R) Core(TM) i7-2700K CPU @ 3.50GHz and 16 Gigabytes RAM.

To allow the reproduction of the experiments, all the code, data and documentation is publicly available [4].

5.3.1 Assumptions

Ideally the traces should have a CPU consumption mean around 36% due to the fact that this is the average mean of CPU usage on servers as shown by Barroso e Holzle (2007), however since there are only a few loads with this characteristic on the Google traces, the traces used were those with CPU load average between 15% and 20% to have the same traces selection criteria on both Planetlab and Google traces.

Unfortunately only the CPU is available on the PlanetLab traces data used on this section. To simulate the other resources, a mapping has been made against the CPU traces. Other resources might have different behavior, especially the network resource that tends to be bursty. Nevertheless, since we are focusing on static allocation techniques, this does not affect the conclusions about the proposed algorithms. This might be not true on dynamic analysis, but this case is out of the scope of this work.

5.3.2 Linear Model Experiments

In this section the results of the experiments using the SPECpower data of a Sugon I840-G25 server are presented. This server has linear power consumption characteristics as presented on Section 5.2.1.

Consumed Power

The Figure 5.4 shows the power consumption comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads on the average case with VMs ranging from 16 to 128. The highlighted blue area on Figure 5.4 represents the power savings comparing the Iterated-KSP strategy (line with circles) with the Energy Unaware strategy (line with dots). The highlighted green area represents the extra power savings from the Iterated-EC strategy (line with squares) in relation to the Iterated-KSP strategy. The dashed line with circles shows the performance of the Iterated-KSP-Mem. The lower the values, the better the strategy, hence the more power savings.

There are significant power savings with the proposed placement strategies compared with the Energy Unaware. The lower the overall cluster load (the less the VMs per physical host) the higher the power savings and consequently the more physical suspended hosts. All the strategies except for the Energy Unaware are able to consume less power due to the fact that they are able to place

[4]https://github.com/vonpupp/dissertation-2014-simulation, Last access on March 15, 2015

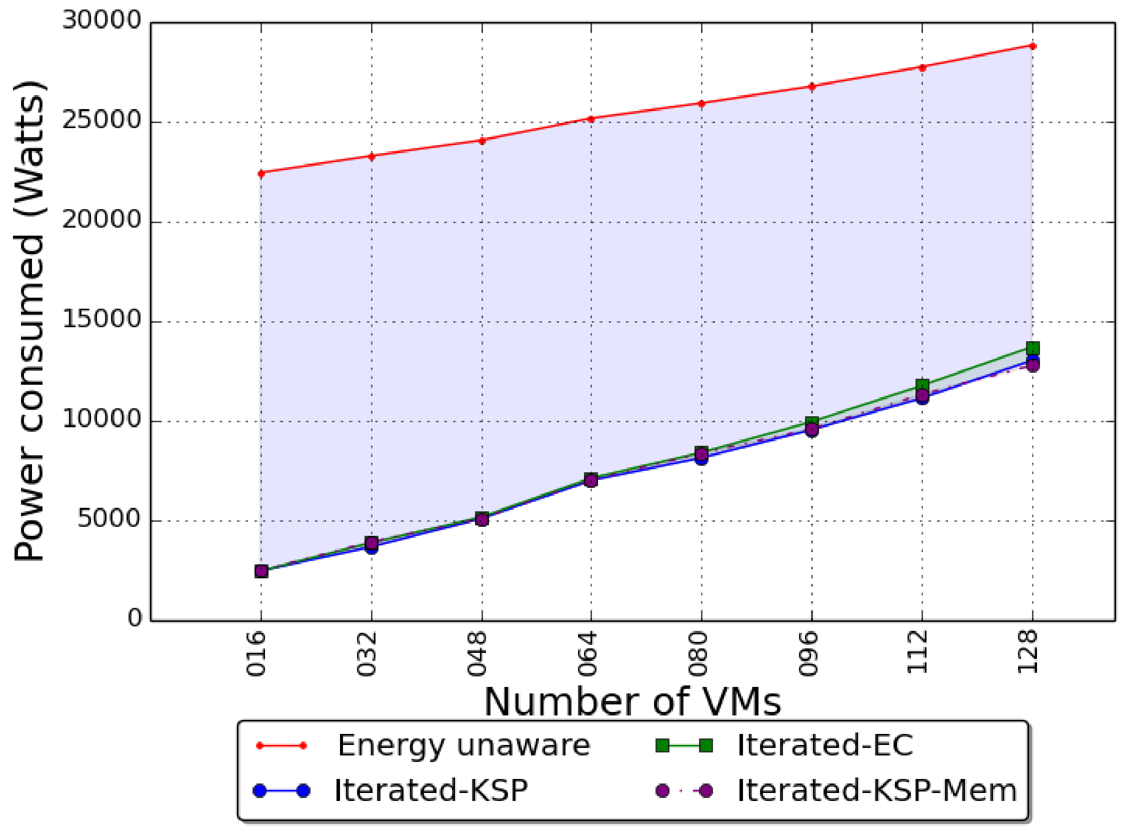


Figure 5.4: *Power consumption comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads. Average case*

more VMs per physical host. Power efficiency resides in the fact that idle hosts are suspended which leverages to an overall power consumption decrease.

In Table 5.10 shows the 95% confidence interval comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads on the average case with VMs ranging from 16 to 128 confidence. The first column represents the number of VMs used for the experiment. Each algorithm is represented by groups, the first group summarizes the Energy Unaware algorithm. In this group is shown the consumed power by that algorithm (W), the standard deviation (SD), the standard error (SE) and the confidence interval (CI).

The second group summarizes the Iterated-KSP algorithm. In this group is only shown the consumed power (W). The rest of the columns were not included since the algorithm is deterministic, this means that for every repetition the algorithm returns the same placement, therefore the SD, SE and CI are always zero.

In a similar way to the first group, the third group summarizes the Iterated-EC algorithm. Finally the fourth group summarize the Iterated-KSP-Mem algorithm. Only the consumed power (W) column was included due to the fact that the algorithm is also deterministic as Iterated-KSP.

	Energy Unaware (W)				*Iterated-KSP* (W)	*Iterated-EC* (W)				*Iterated-KSP-Mem* (W)
VMs	W	SD	SE	CI	W	W	SD	SE	CI	W
16	22,449.06	20.14	3.68	7.52	2,473.40	2,468.59	8.95	1.63	3.34	2,479.00
32	23,285.99	22.12	4.04	8.26	3,674.60	3,878.29	101.41	18.51	37.87	3,922.80
48	24,070.70	19.56	3.57	7.30	5,088.30	5,160.50	111.13	20.29	41.50	5,087.80
64	25,172.55	35.96	6.57	13.43	7,005.60	7,123.50	145.73	26.61	54.42	7,021.40
80	25,934.77	26.92	4.92	10.05	8,139.10	8,405.86	145.35	26.54	54.28	8,375.20
96	26,778.99	30.71	5.61	11.47	9,550.50	9,951.16	108.71	19.85	40.59	9,595.20
112	27,752.24	31.18	5.69	11.64	11,136.80	11,761.59	184.71	33.72	68.97	11,320.10
128	28,841.94	43.40	7.92	16.21	13,026.80	13,709.36	157.02	28.67	58.63	12,776.40

Table 5.10: *Average power consumption and dispersion metrics comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads. 30 repetitions*

Used Hosts

The Table 5.11 shows used hosts comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads on the average case with VMs ranging from 16 to 128. As it can be observed the Iterated-KSP algorithm in general requires less hosts than the other algorithms. The less hosts required to perform the allocation, the more power efficient the algorithm is; which leverages to a higher hardware utilization.

VMs	Energy Unaware *(Hosts)*	Iterated-KSP *(Hosts)*	Iterated-EC *(Hosts)*	Iterated-KSP-Mem *(Hosts)*
16	4.50	4.00	4.00	4.00
32	8.07	6.00	6.80	7.00
48	11.90	9.00	9.37	9.00
64	17.00	13.00	13.70	13.00
80	19.77	15.00	16.33	16.00
96	23.73	18.00	19.70	18.00
112	28.17	21.00	23.80	22.00
128	33.83	25.00	28.03	24.00

Table 5.11: *Used hosts comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads. Average case*

The Figure 5.5 shows the used hosts comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads on the average case with VMs ranging from 16 to 128. The highlighted blue area represents the power savings comparing the Iterated-KSP algorithm (line with circles) with the Energy Unaware algorithm (line with dots). The highlighted green area represents the extra power savings from the Iterated-EC algorithm (line with squares) in relation to the Iterated-KSP algorithm.

We decided not to include the confidence intervals of the used hosts since it is directly related to the consumed power and therefore the confidence intervals behavior is very similarly to the power consumption related confidence intervals.

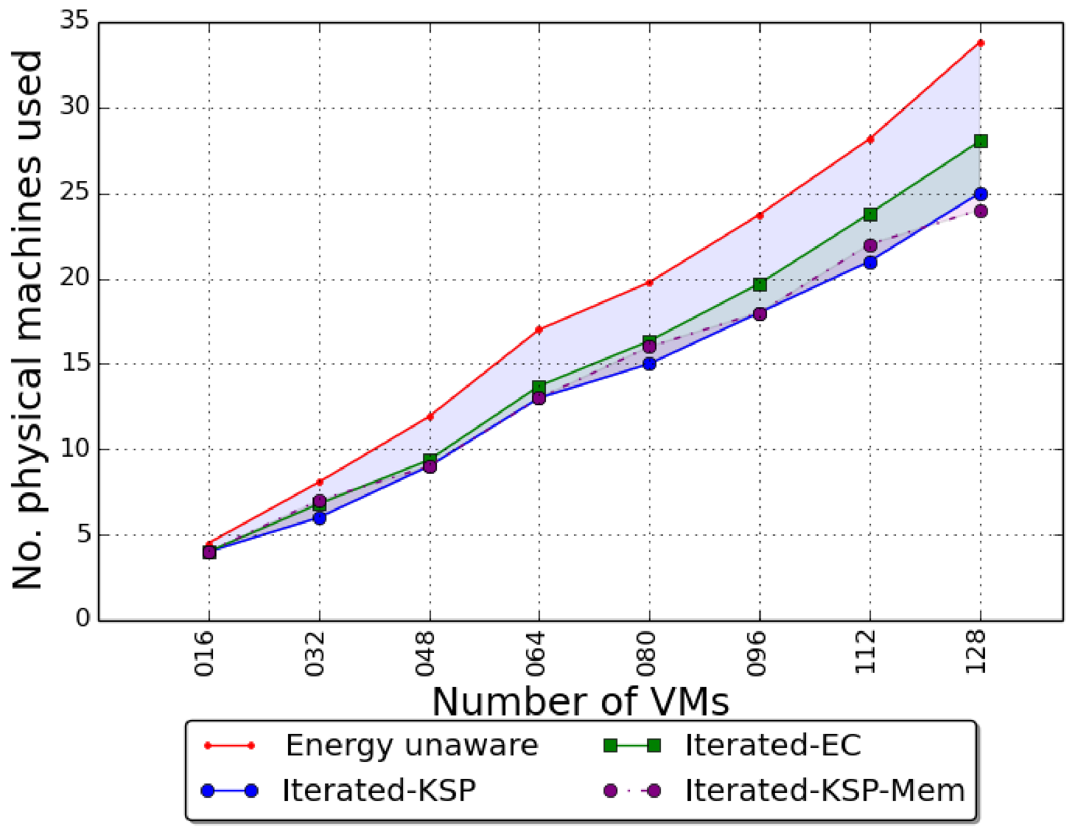


Figure 5.5: *Used hosts comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads. Average case*

As it can be observed the difference of used hosts tend to increase with the increase of the number of VMs. The Iterated-KSP algorithm uses 11.11% to 26.10% less hosts than the Energy Unaware algorithm. As discussed before this is directly related to the power consumption. The Iterated-EC algorithm uses 11.11% to 21.28% less hosts than the Energy Unaware algorithm, while the Iterated-KSP-Mem algorithm uses 11.11% to 29.10%. The fact that the Iterated-KSP-Mem algorithm has a broad range, which translates to less hosts used, is due to the nature of the VMs workloads. VMs that consumes more RAM memory are more efficiently placed using the Iterated-KSP-Mem algorithm.

Idle Hosts

The Table 5.12 shows the idle hosts comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads on the average case with VMs ranging from 16 to 128. As it can be observed except for the Energy Unaware, the algorithms do not allow the hosts to be idle, this means, not to have consumer workloads. All idle hosts are suspended to a lower state of power consumption.

VMs	Energy Unaware *(Hosts)*	Iterated-KSP *(Hosts)*	Iterated-EC *(Hosts)*	Iterated-KSP-Mem *(Hosts)*
16	95.50	0.00	0.00	0.00
32	91.93	0.00	0.00	0.00
48	88.10	0.00	0.00	0.00
64	83.00	0.00	0.00	0.00
80	80.23	0.00	0.00	0.00
96	76.27	0.00	0.00	0.00
112	71.83	0.00	0.00	0.00
128	66.17	0.00	0.00	0.00

Table 5.12: *Idle hosts comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads. Average case*

The Figure 5.6 shows the idle hosts comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads on the average case with VMs ranging from 16 to 128. Note the considerable amount of hosts that are idle, which leads to a very inefficient power utilization placement.

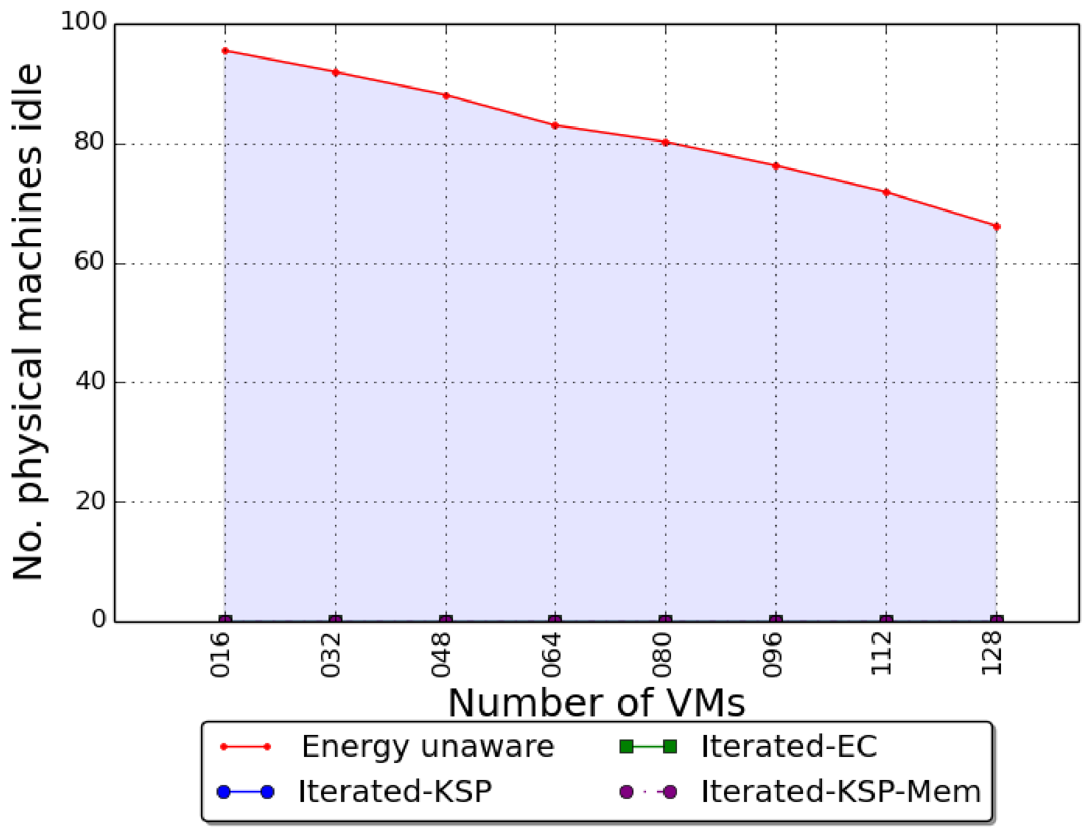


Figure 5.6: *Idle hosts comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads. Average case*

Execution Time

The Table 5.13 shows the execution time comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads on the average case with VMs ranging from 16 to 128.

VMs	Energy Unaware *(Hosts)*	Iterated-KSP *(Hosts)*	Iterated-EC *(Hosts)*	Iterated-KSP-Mem *(Hosts)*
16	$6.21 \cdot 10^{-2}$	0.16	0.43	$4.07 \cdot 10^{-2}$
32	0.19	$7.95 \cdot 10^{-2}$	0.80	0.20
48	0.41	0.14	1.24	0.13
64	0.76	0.34	1.93	0.22
80	1.15	0.42	2.56	0.31
96	1.67	0.53	3.36	0.56
112	2.32	0.69	4.47	0.72
128	3.11	1.00	5.79	1.03

Table 5.13: *Execution time comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads. Average case*

The Figure 5.7 shows the execution time of the algorithms. The Iterated-KSP and Iterated-KSP-Mem are the fastest of the algorithms, this is due to the fact that the Simplex method has been studied for a long time, furthermore, the implementation of the Simplex method use libraries which are highly optimized for this type of calculations. The Energy Unaware algorithm is 52.29% to 268.60% slower than the Iterated-KSP-Mem. The Iterated-KSP algorithm is similar to the Iterated-KSP-Mem in general.

The execution time of the Iterated-EC algorithm is notably higher than the others algorithms. The Iterated-EC algorithm is 304.25% to 949.29% slower than the Iterated-KSP-Mem. This might be due to the parameters used (number of generations and evaluations) which leads to a slower convergence time of the algorithm. The execution time might be improved upon a trade-off with the quality of the solution, which would directly impact the power consumption. Among the experiments performed these parameters were the ones who produced the best results. It is important to highlight that evolutionary computation based algorithms are highly parallelizable by design, therefore it would be possible to optimize execution times but this is out of the scope of this work.

Using non-linear Optimization with Linear SPECpower Models

Additionally it is possible to apply the algorithms using the non-linear heuristics. However, on servers that have a linear SPECpower behavior, they tend to perform worst than the linear optimization heuristics. To show this we are going to summarize only the power consumption obtained results since all the other metrics are directly related to the power consumption.

The Table 5.14 shows the results of applying the Iterated-KSP algorithm comparing the linear and non-linear heuristics using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads on the average case with VMs ranging from 16 to 128. The first column represents the number of VMs used for the experiment. The second column represents the power consumed of the data center with the linear heuristic. Since the SPECpower model used on this experiment

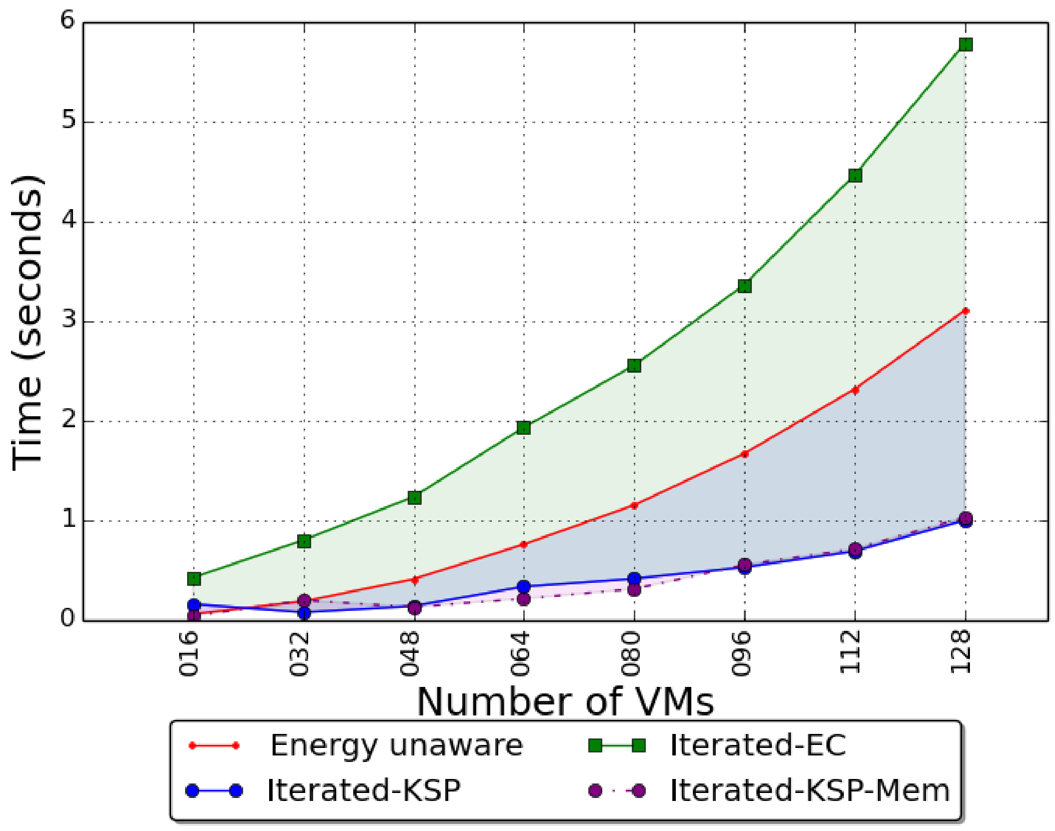


Figure 5.7: *Execution time comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads. Average case*

is a linear model (as described on Section 5.2.1), our non-linear heuristic searches for the highest ssj_ops/watt ratio, which in this case is a load of 100% on the third column; This highest value is referred as the first optimum. The fourth column is the next higher best ssj_ops/watt ratio value, in this case a load of 90%, or second optimum, and so forth with the other columns. All values are expressed in Watts.

VMs	Linear	100%	90%	80%	70%	60%	50%
		1st *opt*	2nd *opt*	3rd *opt*	4th *opt*	5th *opt*	6th *opt*
	(W)	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*
16	2,473.40	2,473.40	2,448.50	2,710.90	2,702.90	2,937.50	3,398.20
32	3,674.60	3,674.60	3,906.00	4,165.20	4,386.90	4,627.80	5,304.60
48	5,088.30	5,088.30	5,046.40	5,556.70	5,778.50	6,250.90	7,156.70
64	7,005.60	7,005.60	6,960.20	7,478.30	7,937.00	8,404.60	9,763.10
80	8,139.10	8,139.10	8,077.20	8,852.40	9,316.40	10,022.40	11,608.80
96	9,550.50	9,550.50	9,753.20	10,073.00	10,766.90	11,710.10	13,055.70
112	11,136.80	11,136.80	11,338.30	11,889.30	12,803.50	13,753.10	15,550.60
128	13,026.80	13,026.80	13,192.60	14,017.40	14,920.70	16,101.80	18,828.20

Table 5.14: *Iterated-KSP algorithm linear vs. non-linear heuristics comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads*

The Table 5.15 shows the difference in Watts based on the previous comparison (Table 5.14). This table shows the difference of each column compared to the linear heuristic. Note that the first optimum of the non-linear optimization behaves as the linear optimization, and therefore all values are zeroes. This is due that in this linear SPECpower model the best ssj_ops/watt ratio is always 100%. The other columns tend to consume more power and therefore the difference result into negative values.

The reason why there are some positives values using 90% target load using 16, 48, 64 and 80 VMs is because this model is discretely linear, this means that it is linear on an interval even when it might not be in general. This might cause some minor variations as observed. More details on can be found on Section 5.2.1.

VMs	Linear	100%	90%	80%	70%	60%	50%
		1st *opt*	2nd *opt*	3rd *opt*	4th *opt*	5th *opt*	6th *opt*
	(W)	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*
16	2,473.40	0.00	24.90	−237.50	−229.50	−464.10	−924.80
32	3,674.60	0.00	−231.40	−490.60	−712.30	−953.20	−1,630.00
48	5,088.30	0.00	41.90	−468.40	−690.20	−1,162.60	−2,068.40
64	7,005.60	0.00	45.40	−472.70	−931.40	−1,399.00	−2,757.50
80	8,139.10	0.00	61.90	−713.30	−1,177.30	−1,883.30	−3,469.70
96	9,550.50	0.00	−202.70	−522.50	−1,216.40	−2,159.60	−3,505.20
112	11,136.80	0.00	−201.50	−752.50	−1,666.70	−2,616.30	−4,413.80
128	13,026.80	0.00	−165.80	−990.60	−1,893.90	−3,075.00	−5,801.40

Table 5.15: *Linear vs. non-linear heuristics difference using the Iterated-KSP algorithm comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads*

Likewise, the Table 5.16 shows the results of applying the Iterated-EC algorithm comparing the linear and non-linear heuristics using 100 Sugon servers with a linear SPECpower model and

PlanetLab workloads on the average case with VMs ranging from 16 to 128. In a similar way, the first column represents the number of VMs used for the experiments. The second column represents the linear heuristic. The third column shows a load of 100%, also referred to as the first optimum. The fourth column is the next best ssj_ops/watt ratio value, in this case, a load of 90% of the load and so forth with the other columns. All values are expressed in Watts.

VMs	Linear	100%	90%	80%	70%	60%	50%
		1st *opt*	2nd *opt*	3rd *opt*	4th *opt*	5th *opt*	6th *opt*
	(W)	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*
16	2,468.59	2,467.71	2,460.89	2,701.16	2,742.83	2,992.00	3,202.17
32	3,878.29	3,886.09	3,906.37	4,086.25	4,349.83	4,673.39	5,207.59
48	5,160.50	5,181.83	5,191.31	5,396.31	5,772.11	6,277.42	6,982.65
64	7,123.50	7,114.16	7,126.82	7,422.28	7,852.24	8,453.10	9,452.76
80	8,405.86	8,465.77	8,471.75	8,749.01	9,304.13	10,039.20	11,163.70
96	9,951.16	9,953.95	10,009.93	10,319.12	10,909.23	11,791.26	13,095.94
112	11,761.59	11,791.73	11,883.30	12,230.27	12,884.48	13,960.06	15,616.57
128	13,709.36	13,674.12	13,953.76	14,406.83	15,309.00	16,544.62	18,262.87

Table 5.16: *Linear vs. non-linear heuristics using the Iterated-EC algorithm comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads*

VMs	Linear	100%	90%	80%	70%	60%	50%
		1st *opt*	2nd *opt*	3rd *opt*	4th *opt*	5th *opt*	6th *opt*
	(W)	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*
16	2,468.59	0.88	7.70	−232.57	−274.24	−523.41	−733.58
32	3,878.29	−7.80	−28.08	−207.96	−471.54	−795.10	−1,329.30
48	5,160.50	−21.33	−30.81	−235.81	−611.61	−1,116.92	−1,822.15
64	7,123.50	9.34	−3.33	−298.78	−728.75	−1,329.60	−2,329.26
80	8,405.86	−59.90	−65.89	−343.15	−898.26	−1,633.34	−2,757.84
96	9,951.16	−2.79	−58.76	−367.96	−958.06	−1,840.10	−3,144.78
112	11,761.59	−30.14	−121.71	−468.68	−1,122.89	−2,198.47	−3,854.98
128	13,709.36	35.24	−244.40	−697.47	−1,599.64	−2,835.26	−4,553.51

Table 5.17: *Linear vs. non-linear heuristics difference using the Iterated-EC algorithm comparison using 100 Sugon servers with a linear SPECpower model and PlanetLab workloads*

The Table 5.17 shows the difference in Watts based on the previous comparison (Table 5.16). This table shows the difference of each column compared to the linear heuristic.

Note that the first optimum of the non-linear optimization behaves as the linear optimization. This is due that in this linear SPECpower model the best ssj_ops/watt ratio is always 100%. The other columns tend to consume more power and therefore the difference result into negative values. In this case the first optimum of the non-linear optimization is similar to the linear optimization nevertheless they are not the same. This is due to the fact that the Iterated-EC algorithm relies on an n-point crossover and therefore the solutions are not exactly the same on every repetition.

The reason why the 16VMs experiment has a positive values using 90% target load is because this model is discretely linear. This means that it is linear on an interval even when it might not be in general. This might cause some minor variations as observed. More details on can be found on Section 5.2.1.

5.3.3 Non-Linear Model Experiments

As introduced before, it is possible to use non-linear heuristics to optimize the placement. This section presents the results of the experiments using the SPECpower data of a Fujitsu PRIMERGY RX1330 M1 server which has non-linear power consumption characteristics as introduced on Section 5.2.2.

In this subsection only the power consumption results are presented since all the other metrics are directly related to the power consumption.

The Table 5.18 shows the results of applying the Iterated-KSP algorithm comparing the linear and non-linear heuristics using 100 Fujitsu servers with a non-linear SPECpower model and PlanetLab workloads on the average case with VMs ranging from 16 to 128. The first column represents the number of VMs used for the experiment. The second column represents the linear heuristic. Since the SPECpower model used on this experiment is a non-linear model (as described on Section 5.2.2), the non-linear heuristic searches for the highest ssj_ops/watt ratio, which in this case is a target load of 70%; we refer to this highest value as the first optimum, which is on the sixth column. The next highest best ssj_ops/watt ratio value, in this case targeting 60%, or second optimum is shown on the seventh column. The next highest best ssj_ops/watt ratio value, targeting 80% is shown on the fifth column. The next highest best ssj_ops/watt ratio value, targeting 90% is shown on the fourth column.

VMs	Linear	100%	90%	80%	70%	60%	50%
		5th *opt*	4th *opt*	3rd *opt*	1st *opt*	2nd *opt*	6th *opt*
	(W)	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*
16	674.64	674.64	675.32	671.76	665.18	668.82	692.12
32	806.93	806.93	794.29	794.49	787.03	786.16	819.44
48	940.71	940.71	911.86	904.26	896.87	900.10	942.73
64	1,106.75	1,106.75	1,071.23	1,066.91	1,047.35	1,053.42	1,117.83
80	1,222.52	1,222.52	1,190.85	1,170.09	1,155.88	1,163.05	1,242.37
96	1,374.64	1,374.64	1,317.76	1,286.43	1,270.24	1,279.37	1,348.63
112	1,511.72	1,511.72	1,457.15	1,427.30	1,417.54	1,425.16	1,511.98
128	1,684.50	1,684.50	1,622.66	1,581.77	1,574.31	1,584.97	1,720.26

Table 5.18: *Iterated-KSP algorithm linear vs. non-linear heuristics comparison using 100 Fujitsu servers with a non-linear SPECpower model and PlanetLab workloads*

The Table 5.19 shows the difference in Watts based on the previous comparison (Table 5.18). This table shows the difference of each column compared to the linear heuristic. Note that the fifth optimum of the non-linear optimization behaves as the linear optimization, and therefore all values are zeroes (as shown on the third column). It is interesting to notice that positive values represents power savings, therefore the column with the highest values is the best heuristic. In this case, the best heuristic is the one targeting 70% load. The next best heuristic is the on targeting 60% load, except for the 128 VMs scenario which is outperformed by the heuristic targeting 80% load. There is an expected behavior on non-linear heuristics however there might be exceptions as this one. This is because of the global optimum phenomena detailed on the Section 5.2.2.

VMs	Linear	100%	90%	80%	70%	60%	50%
		$5^{th}opt$	$4^{th}opt$	$3^{rd}opt$	$1^{st}opt$	$2^{nd}opt$	$6^{th}opt$
	(W)	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*
16	674.64	0.00	−0.68	2.88	9.46	5.82	−17.48
32	806.93	0.00	12.64	12.44	19.90	20.77	−12.51
48	940.71	0.00	28.85	36.45	43.84	40.61	−2.02
64	1,106.75	0.00	35.52	39.84	59.40	53.33	−11.08
80	1,222.52	0.00	31.67	52.43	66.64	59.47	−19.85
96	1,374.64	0.00	56.88	88.21	104.40	95.27	26.01
112	1,511.72	0.00	54.57	84.42	94.18	86.56	−0.26
128	1,684.50	0.00	61.84	102.73	110.19	99.53	−35.76

Table 5.19: *Linear vs. non-linear heuristics difference using the Iterated-KSP algorithm comparison using 100 Fujitsu servers with a non-linear SPECpower model and PlanetLab workloads*

5.4 Experiments Using Workloads from Google

Google made publicly available some workloads from one of its computing clusters as part of the Google Cluster Data project. These workloads represent 29 days of data from May 2011 on a cluster of 11,000 machines and they are divided in 500 parts, from 00000 to 00499. The workloads are about 40GB of *compressed* text data in Comma Separated Value (CSV) format. The tasks are executed during a variable *units of time*. Some tasks consume such a small amount of resources that are not accounted as a whole unit of CPU, RAM, Disk I/O or Disk space Reiss *et al.* (2011) while other tasks are really long, including tasks running for the whole period of time (29 days).

The workloads are composed by various *jobs*. Each job is created by an *user*, and it is comprised of one or more *tasks*, including the requirements used for scheduling the tasks on physical machines. There is no known distribution that fits the data, nevertheless the resources appear to form a long-tailed distribution Reiss *et al.* (2011).

In the simulations we considered a scenario of a Cloud IaaS using 130 hosts with VMs allocation requests varying from 16 to 128 VMs with 16 increments. Two types of hosts were considered. One type of server was the Sugon I840-G25 server which has linear power consumption characteristics as presented on Section 5.2.1. The Section 5.4.2 is dedicated to discuss the results of the experiments when all the hosts were simulated as being this server. The other type of server was the Fujitsu PRIMERGY RX1330 M1 server which has non-linear power consumption characteristics as presented on Section 5.2.2. The Section 5.4.3 is dedicated to discuss the results of the experiments when all the hosts were simulated as being this server.

Each simulation was repeated 30 times to check if there was a clear tendency, and later the data was reduced to three cases: best, worst and average case (mean). The simulations were executed using the same host where the PlanetLab experiment were executed.

5.4.1 Assumptions

Google does not provide data about their computing environment and the data itself is anonymized, therefore it is not possible to know if the application is running on a VM, a container or even on the real OS. Nevertheless after a public online discussion held with one of the researchers from Google, a container based virtualization assumption can be made [5]. Each job has a multitude of tasks, that may or not run in the same machine. However, a task is a single Linux program, possibly consisting of multiple processes, to be run on a single machine Reiss *et al.* (2011). So it is reasonable to consider each task as a single VM instance or a container.

Another assumption is about machine resources. Each resource (including CPU, memory, disk space and I/O time) is normalized, relative to the largest capacity of the resource on any machine in the trace Reiss *et al.* (2011).

The whole workload including the 500 parts resulted in a database of almost 190GB with 24, 281, 242 unique tasks. Unfortunately the disk time measurement is only included in the first 14 days, because of a change in the monitoring system. Therefore only the first 50 parts, from 00000 to 00049 has been considered, to be able to have disk I/O time data. This is equivalent to 2, 713, 386 unique tasks. About 1% of the jobs in the trace have missing values (Not A Number, or NAN), therefore these records have been excluded from the analysis. Another omission from the trace is any information about network activity.

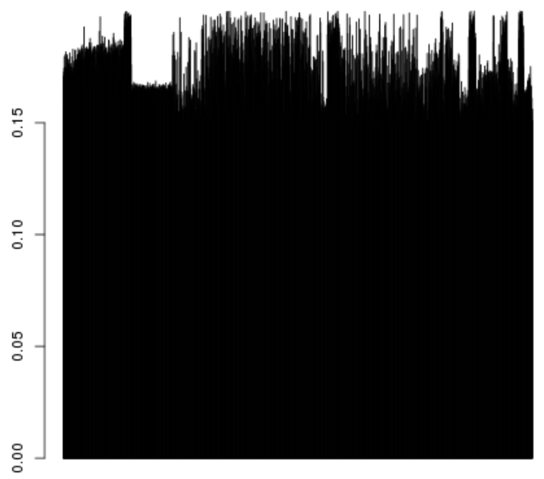
The traces used for the experiments had a CPU load average between 15% to 20%, to get only tasks that had relevant CPU usage. This is equivalent to 2, 762 unique tasks. A selection of the first 128 tasks has been made to represent the VMs for the simulations, to maintain consistency with the prior experiments with PlanetLab traces. It worth noticing that according to Reiss *et al.* (2012) there is no known distribution that seems to approximate properly the Google workloads. This might be due to the unpredictability on consumer applications due to human behavior, and the large quantity of heterogeneous data.

The characteristics of the CPU usage for the selected tasks are shown in Figure 5.8. The majority of the tasks, approximately 74.22% of the selected tasks or about 2350 tasks, are between 15%-18% range of average CPU usage. The rest of tasks, or approximately 25.78% of selected tasks, are above 18% of average CPU usage. Note that there is no known distribution that seems to approximate properly, and the more tasks included, the more it behaves as a long-tail distribution.

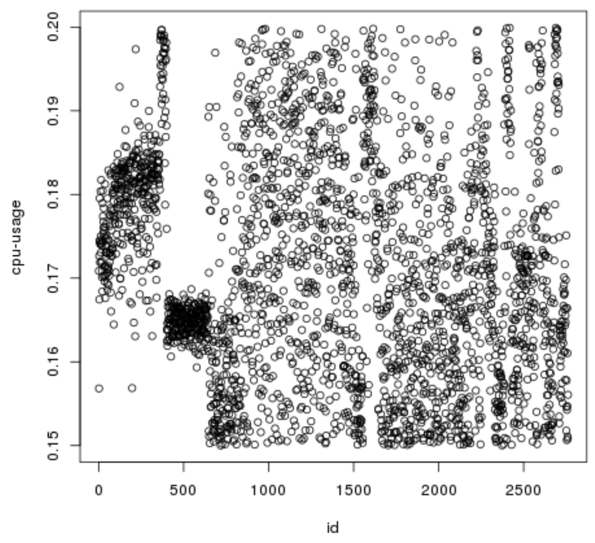
The characteristics of the RAM memory usage for the selected tasks are shown in Figure 5.9. The majority of the tasks, about 94.13% of the selected tasks or approximately 2600 tasks, are in the 0%-20% range of average memory usage. The rest of the tasks, or approximately 5.87% or about 162 tasks, are above 20%. These characteristics are similar to a long-tail distribution.

The characteristics of the disk I/O time for the selected tasks are shown in Figure 5.10. The majority of the tasks, about 65.17% of the selected tasks or approximately 1800 tasks, are in the 0%-0.5% range of average disk time. The rest of the tasks, or approximately 34.84% or about 962

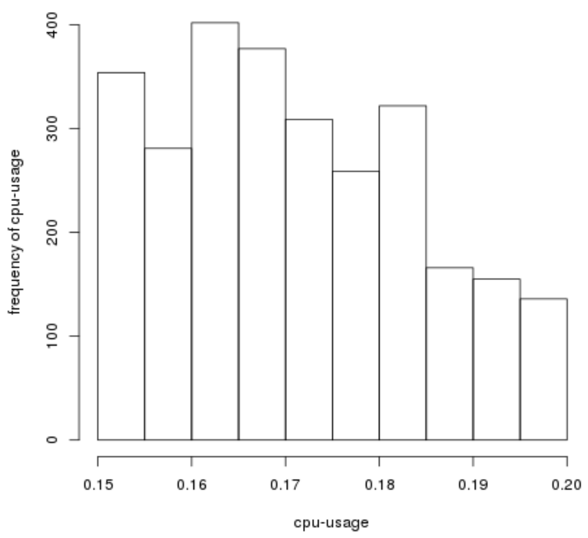
[5]https://groups.google.com/d/msg/googleclusterdata-discuss/ojH3KEx0Pe4/Rogl7sf6Lg0J



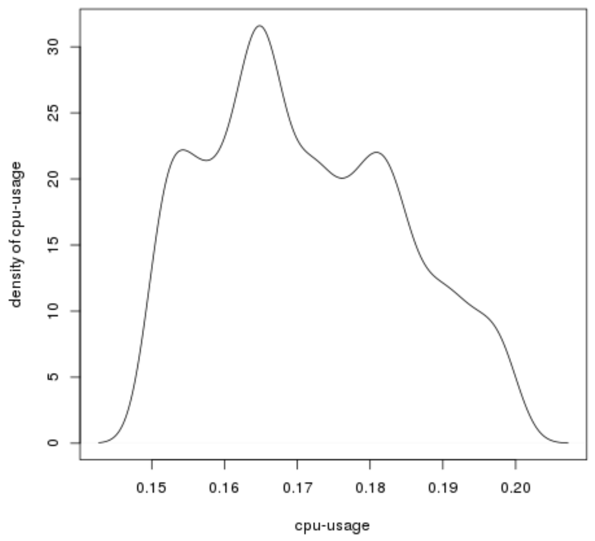
(a) *Bar plot of cpu-usage*



(b) *Plot of cpu-usage*



(c) *Histogram of cpu-usage*

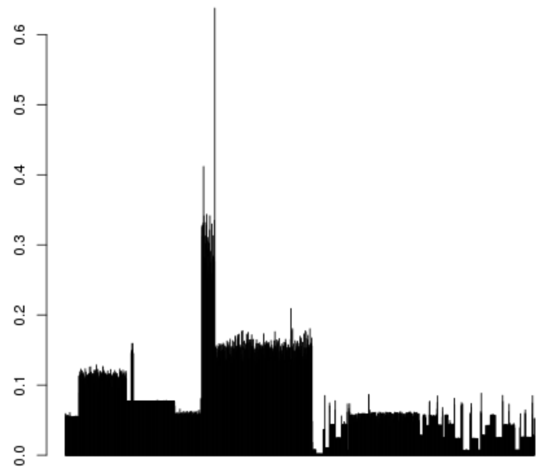


(d) *Density of cpu-usage*

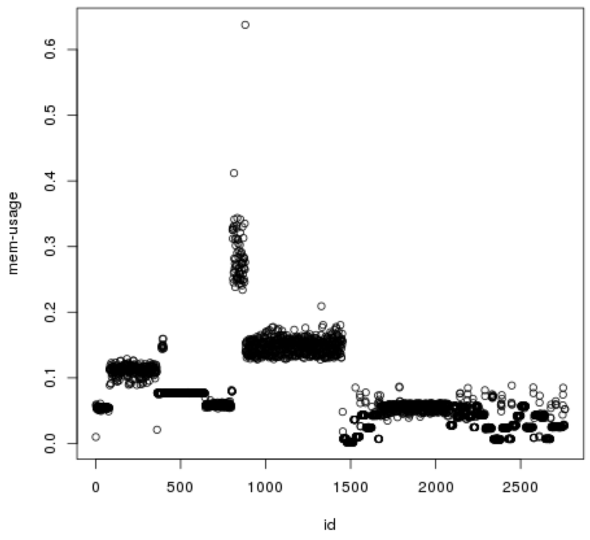
Figure 5.8: *Statistical plots of cpu-usage*

tasks, are above 0.5%. These characteristics are similar to a long-tail distribution. Note that it seems to exist another group of tasks that consumes more disk-time, around 0.15%, perhaps these are I/O intensive tasks, nevertheless the I/O time is in general always less than 0.20%.

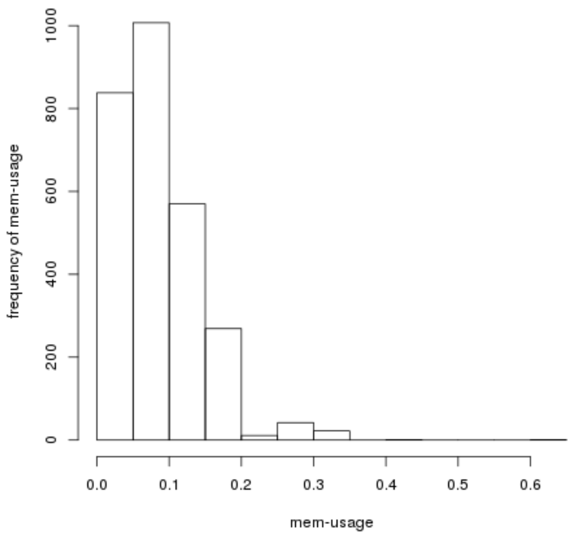
The characteristics of the disk usage for the selected tasks are shown in Figure 5.11. The majority of the tasks, about 85.80% of the selected tasks or approximately 2370 tasks, are in the 0%-0.05% range of average disk usage. The rest of the tasks, or approximately 14.20% or about 392 tasks, are above 0.05%. These characteristics are similar to a long-tail distribution.



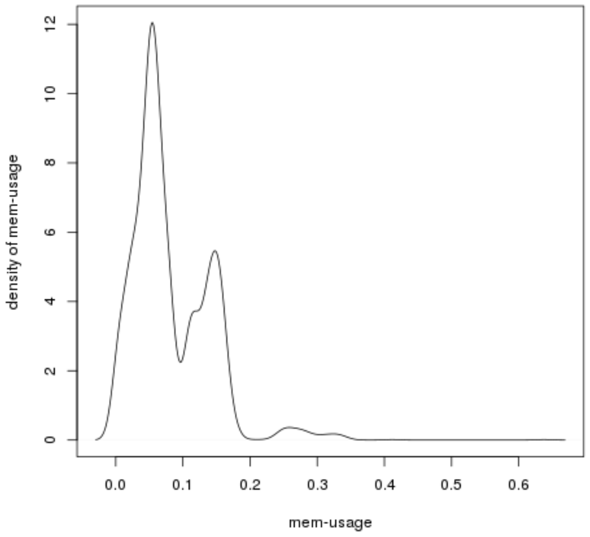
(a) *Bar plot of mem-usage*



(b) *Plot of mem-usage*



(c) *Histogram of mem-usage*



(d) *Density of mem-usage*

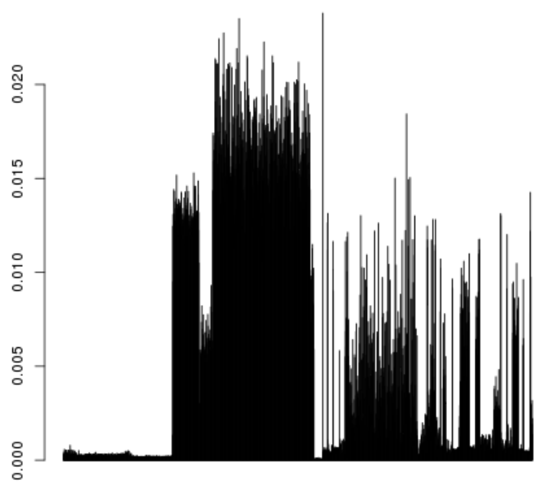
Figure 5.9: *Statistical plots of mem-usage*

5.4.2 Linear Model Experiments

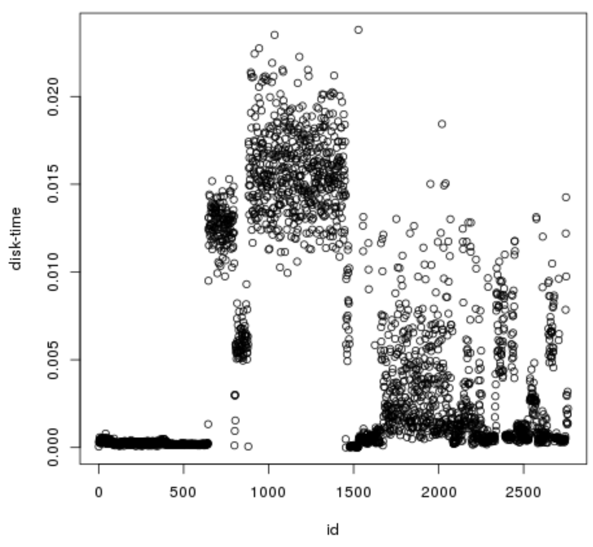
In this section the results of the experiments using the SPECpower data of a Sugon I840-G25 server are discussed. This server has linear power consumption characteristics as presented on Section 5.2.1. The power consumption comparisons will be the only results discussed in this section. The rest of the metrics will not be included due to the similarity with the Section 5.3.2.

Consumed Power

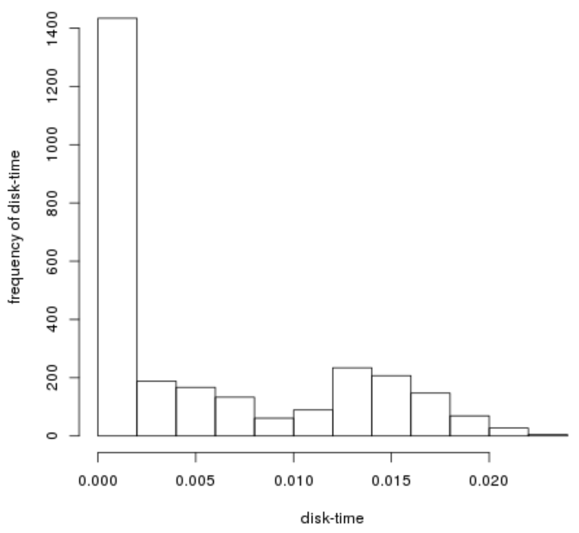
The Figure 5.12 shows the power consumption comparison using 130 Sugon servers with a linear SPECpower model and Google workloads on the average case with VMs ranging from 16 to 128.



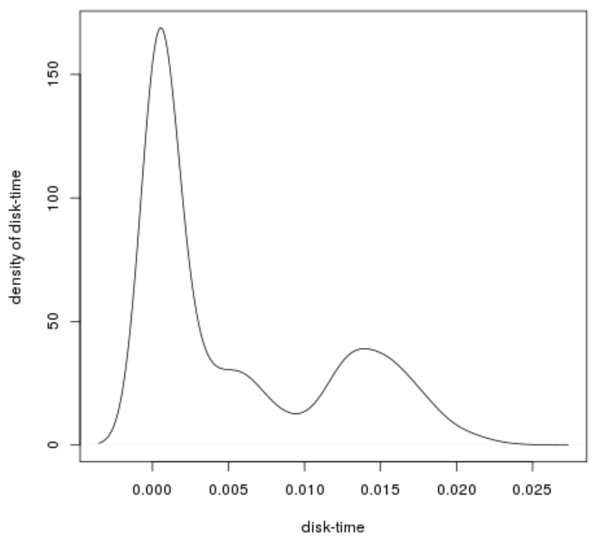
(a) *Bar plot of disk-time*



(b) *Plot of disk-time*



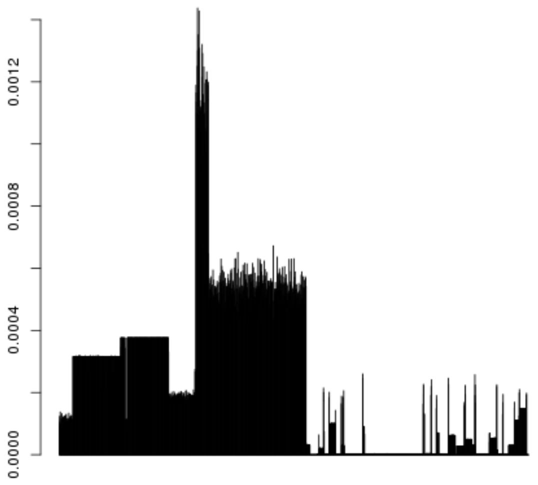
(c) *Histogram of disk-time*



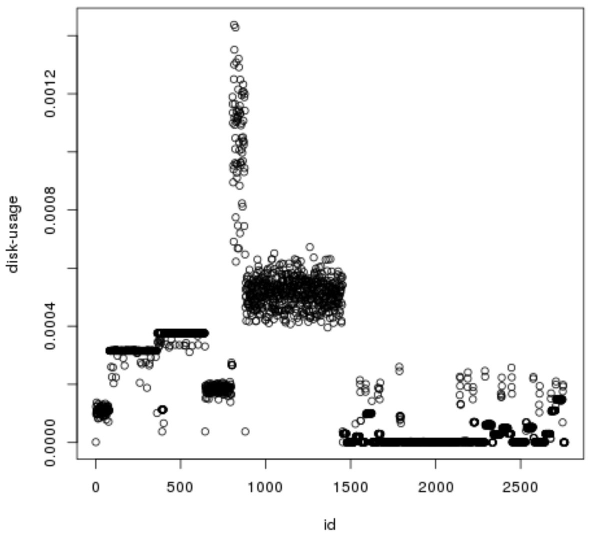
(d) *Density of disk-time*

Figure 5.10: *Statistical plots of disk-time*

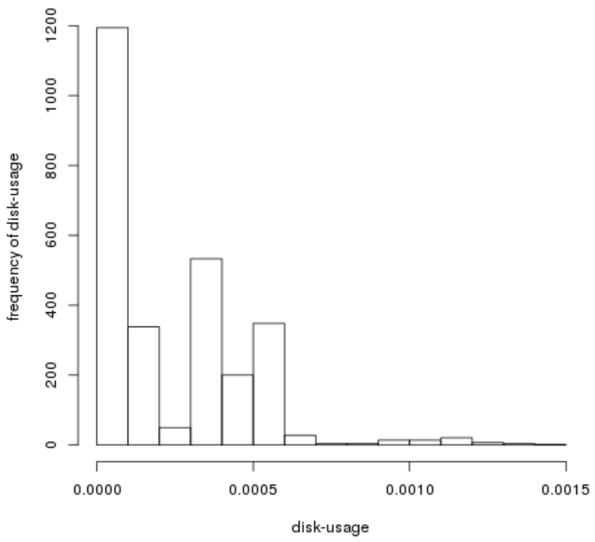
The highlighted blue area on Figure 5.12 represents the power savings comparing the Iterated-KSP strategy (line with circles) with the Energy Unaware strategy (line with dots). The highlighted green area represents the extra power savings from the Iterated-EC strategy (line with squares) in relation to the Iterated-KSP strategy. The dashed line with circles shows the performance of the Iterated-KSP-Mem. The lower the values, the better the strategy, hence the more power savings.



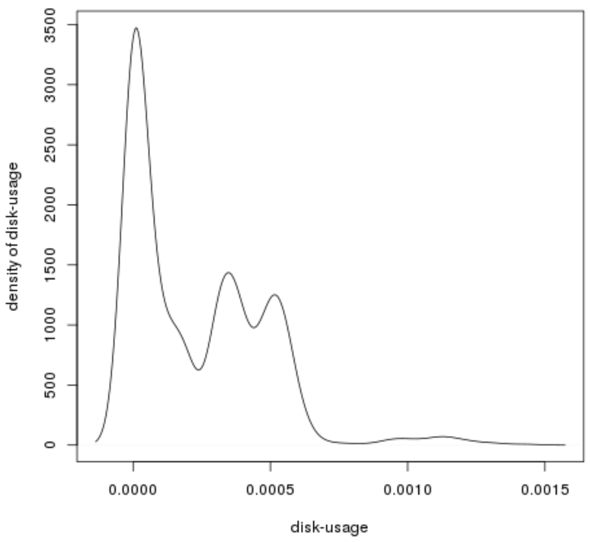
(a) *Bar plot of disk-usage*



(b) *Plot of disk-usage*



(c) *Histogram of disk-usage*



(d) *Density of disk-usage*

Figure 5.11: *Statistical plots of disk-usage*

VMs	Energy Unaware *(W)*	Iterated-KSP *(W)*	Iterated-EC *(W)*	Iterated-KSP-Mem *(W)*
16	29,210.57	3,214.54	3,210.32	3,214.54
32	30,811.95	5,853.38	5,870.51	5,853.38
48	32,428.60	8,725.52	8,718.53	8,725.52
64	34,032.80	11,374.40	11,361.22	11,374.40
80	35,630.50	14,010.70	14,018.38	14,010.70
96	37,237.02	16,866.58	16,860.19	16,866.58
112	38,853.96	19,536.25	19,516.15	19,536.25
128	40,488.08	22,210.13	22,284.08	22,210.13

Table 5.20: *Power consumption comparison using 130 Sugon servers with a linear SPECpower model and Google workloads. Average case*

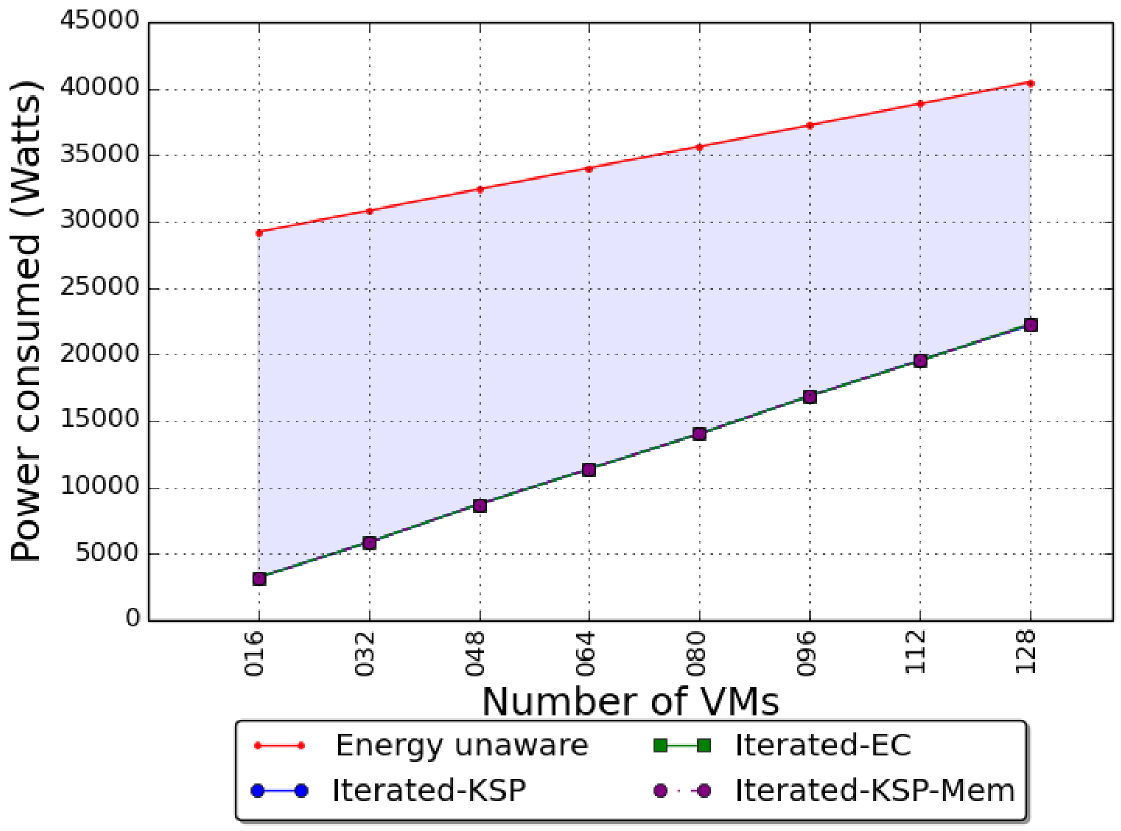


Figure 5.12: *Power consumption comparison using 130 Sugon servers with a linear SPECpower model and Google workloads. Average case*

As it can be seen there are significant power savings with the proposed placement strategies compared with the Energy Unaware. The lower the overall cluster load (the less the VMs per physical host) the higher the power savings and consequently the more physical suspended hosts. All the strategies except for the Energy Unaware are able to consume less power due to the fact that they are able to place more VMs per physical host. Power efficiency resides in the fact that idle hosts are suspended which leverages to an overall power consumption decrease.

The Table 5.21 shows the 95% confidence interval comparison using 130 Sugon servers with a linear SPECpower model and Google workloads on the average case with VMs ranging from 16 to 128 confidence. The first column represents the number of VMs used for the experiment. Each algorithm is represented by groups, the first group summarizes the Energy Unaware algorithm. In this group is shown the consumed power by that algorithm (W), the standard deviation (SD), the standard error (SE) and the confidence interval (CI).

The second group summarizes the Iterated-KSP algorithm. In this group is only shown the consumed power (W). The rest of the columns were not included since the algorithm is deterministic, this means that for every repetition the algorithm returns the same placement, therefore SD, SE and CI are always zero.

In a similar way to the first group, the third group summarizes the Iterated-EC algorithm. Finally the fourth group summarize the Iterated-KSP-Mem algorithm. Only the consumed power (W) column has been included due to the fact that the algorithm is also deterministic as Iterated-

KSP.

	Energy Unaware (W)				*Iterated-KSP* (W)	*Iterated-EC* (W)				*Iterated-KSP-Mem* (W)
VMs	W	SD	SE	CI	W	W	SD	SE	CI	W
16	29,210.57	8.06	1.47	3.01	3,214.54	3,210.32	7.19	1.31	2.68	3,214.54
32	30,811.95	8.76	1.60	3.27	5,853.38	5,870.51	68.57	12.52	25.60	5,853.38
48	32,428.60	5.85	1.07	2.18	8,725.52	8,718.53	4.12	0.75	1.54	8,725.52
64	34,032.80	7.96	1.45	2.97	11,374.40	11,361.22	6.28	1.15	2.35	11,374.40
80	35,630.50	11.35	2.07	4.24	14,010.70	14,018.38	69.16	12.63	25.82	14,010.70
96	37,237.02	7.57	1.38	2.83	16,866.58	16,860.19	43.54	7.95	16.26	16,866.58
112	38,853.96	10.55	1.93	3.94	19,536.25	19,516.15	39.40	7.19	14.71	19,536.25
128	40,488.08	12.54	2.29	4.68	22,210.13	22,284.08	114.19	20.85	42.64	22,210.13

Table 5.21: *Average power consumption and dispersion metrics comparison using 130 Sugon servers with a linear SPECpower model and Google workloads. 30 repetitions*

5.4.3 Non-Linear Model Experiments

As introduced before, it is possible to use non-linear heuristics to optimize the placement. This section presents the results of the experiments using the SPECpower data of a Fujitsu PRIMERGY RX1330 M1 server which has non-linear power consumption characteristics as introduced on Section 5.2.2.

In non-linear scenarios, the non-linear heuristics tend to perform better than the linear optimization heuristics. To show this only the power consumption figures are summarized since all the other metrics are directly related to the power consumption.

The Table 5.22 shows the results of applying the Iterated-KSP algorithm comparing the linear and non-linear heuristics using 130 Sugon servers with a linear SPECpower model and Google workloads on the average case with VMs ranging from 16 to 128. The first column represents the number of VMs used for the experiment. The second column represents the linear heuristic. Since the SPECpower model used on this experiment is a non-linear model (as described on Section 5.2.2), the non-linear heuristic searches for the highest ssj_ops/watt ratio, which in this case is a target load of 70%; we refer to this highest value as the first optimum, which is on the sixth column. The next highest best ssj_ops/watt ratio value, in this case targeting 60%, or second optimum is shown on the seventh column. The next highest best ssj_ops/watt ratio value, targeting 80% is shown on the fifth column. The next highest best ssj_ops/watt ratio value, targeting 90% is shown on the fourth column.

Note that on this specific SPECpower model the next ssj_ops/watt ratio value, targets 100%, which is equivalent to apply the linear optimization heuristics, which maximizes the number of VMs per hosts and minimizes the number of hosts used. This value is shown on the third column. The next highest best ssj_ops/watt ratio value, targeting 50% is shown on the eight column. All values are expressed in Watts.

VMs	Linear	100%	90%	80%	70%	60%	50%
		5th *opt*	4th *opt*	3rd *opt*	1st *opt*	2nd *opt*	6th *opt*
	(W)	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*
16	890.97	890.97	881.51	869.04	867.85	874.82	948.21
32	1,138.49	1,138.49	1,134.16	1,099.52	1,105.37	1,105.37	1,280.00
48	1,391.22	1,391.22	1,375.30	1,329.53	1,334.59	1,334.59	1,611.34
64	1,633.39	1,633.39	1,624.71	1,560.80	1,566.08	1,566.86	1,943.39
80	1,880.26	1,880.26	1,876.40	1,790.06	1,796.69	1,796.69	2,274.37
96	2,127.50	2,127.50	2,111.83	2,017.87	2,021.56	2,021.56	2,603.89
112	2,376.44	2,376.44	2,367.78	2,254.51	2,258.21	2,258.21	2,938.46
128	2,637.31	2,637.31	2,633.71	2,496.34	2,499.98	2,499.98	3,275.50

Table 5.22: *Iterated-KSP algorithm linear vs. non-linear heuristics comparison using 130 Fujitsu servers with a non-linear SPECpower model and PlanetLab workloads*

The Table 5.23 shows the difference in Watts based on the previous comparison (Table 5.22). This table shows the difference of each column compared to the linear heuristic. Note that the fifth optimum of the non-linear optimization behaves as the linear optimization, and therefore all values are zeroes (as shown on the third column). It is interesting to notice that positive values represents power savings, therefore the column with the highest values is the best heuristic. In this case, the best heuristic is the one targeting 80% load. The next best heuristic is the one targeting 70% load.

VMs	Linear	100%	90%	80%	70%	60%	50%
		5th *opt*	4th *opt*	3rd *opt*	1st *opt*	2nd *opt*	6th *opt*
	(W)	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*
16	890.97	0.00	9.46	21.93	23.13	16.16	−57.24
32	1,138.49	0.00	4.33	38.96	33.11	33.11	−141.51
48	1,391.22	0.00	15.92	61.69	56.63	56.63	−220.12
64	1,633.39	0.00	8.69	72.59	67.32	66.53	−310.00
80	1,880.26	0.00	3.86	90.20	83.57	83.57	−394.11
96	2,127.50	0.00	15.68	109.64	105.95	105.95	−476.38
112	2,376.44	0.00	8.66	121.93	118.23	118.23	−562.02
128	2,637.31	0.00	3.59	140.96	137.33	137.33	−638.20

Table 5.23: *Linear vs. non-linear heuristics difference using the Iterated-KSP algorithm comparison using 130 Fujitsu servers with a non-linear SPECpower model and PlanetLab workloads*

The Table 5.24 shows the results of applying the Iterated-EC algorithm comparing the linear and non-linear heuristics using 130 Sugon servers with a linear SPECpower model and Google workloads on the average case with VMs ranging from 16 to 128. The first column represents the number of VMs used for the experiment. The second column represents the linear heuristic. Since the SPECpower model used on this experiment is a non-linear model (as described on Section 5.2.2), the non-linear heuristic searches for the highest ssj_ops/watt ratio, which in this case is a target load of 70%; we refer to this highest value as the first optimum, which is on the sixth column. The next highest best ssj_ops/watt ratio value, in this case targeting 60%, or second optimum is shown on the seventh column. The next highest best ssj_ops/watt ratio value, targeting 80% is shown on the fifth column. The next highest best ssj_ops/watt ratio value, targeting 90% is shown on the fourth column.

Note that on this specific SPECpower model the next ssj_ops/watt ratio value, targets 100%, which is equivalent to apply the linear optimization heuristics, which maximizes the number of VMs per hosts and minimizes the number of hosts used. This value is shown on the third column. The next highest best ssj_ops/watt ratio value, targeting 50% is shown on the eight column. All values are expressed in Watts.

VMs	Linear	100%	90%	80%	70%	60%	50%
		$5^{th}opt$	$4^{th}opt$	$3^{rd}opt$	$1^{st}opt$	$2^{nd}opt$	$6^{th}opt$
	(W)	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*
16	888.47	888.10	881.87	869.84	868.72	874.20	948.21
32	1,138.73	1,137.80	1,134.07	1,101.94	1,102.07	1,107.28	1,279.77
48	1,390.87	1,390.66	1,373.61	1,332.97	1,332.79	1,338.57	1,611.07
64	1,631.14	1,631.20	1,624.78	1,565.54	1,565.24	1,571.12	1,943.51
80	1,879.41	1,879.41	1,870.06	1,797.90	1,798.09	1,801.64	2,274.95
96	2,124.67	2,125.30	2,109.46	2,029.43	2,028.66	2,032.93	2,604.02
112	2,373.59	2,372.84	2,366.22	2,265.30	2,264.83	2,267.95	2,940.74
128	2,634.69	2,633.48	2,623.24	2,506.05	2,507.09	2,507.94	3,276.05

Table 5.24: *Iterated-EC algorithm linear vs. non-linear heuristics comparison using 130 Fujitsu servers with a non-linear SPECpower model and PlanetLab workloads*

The Table 5.25 shows the difference in Watts based on the previous comparison (Table 5.24). This table shows the difference of each column compared to the linear heuristic. Note that the fifth optimum of the non-linear optimization behaves as the linear optimization. In this case the values are not always zeroes as in the Iterated-KSP, nevertheless they are near to zero in most of the cases (as shown on the third column). It is interesting to notice that positive values represents power savings, therefore the column with the highest values is the best heuristic. In this case, the best heuristic is between 70% and 80% load depending on the quantity of VMs. This might be due to the evaluation function used that directly relates to the convergence of the heuristics.

VMs	Linear	100%	90%	80%	70%	60%	50%
		5th *opt*	4th *opt*	3rd *opt*	1st *opt*	2nd *opt*	6th *opt*
	(W)	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*	*(W)*
16	888.47	0.37	6.60	18.63	19.75	14.27	−59.74
32	1,138.73	0.92	4.66	36.79	36.66	31.45	−141.04
48	1,390.87	0.21	17.26	57.90	58.08	52.30	−220.20
64	1,631.14	$-6.31 \cdot 10^{-2}$	6.37	65.60	65.90	60.02	−312.37
80	1,879.41	$-9.87 \cdot 10^{-4}$	9.35	81.52	81.32	77.77	−395.54
96	2,124.67	−0.63	15.21	95.24	96.01	91.74	−479.35
112	2,373.59	0.75	7.37	108.29	108.76	105.64	−567.15
128	2,634.69	1.21	11.46	128.65	127.60	126.75	−641.36

Table 5.25: *Linear vs. non-linear heuristics difference using the Iterated-EC algorithm comparison using 130 Fujitsu servers with a non-linear SPECpower model and PlanetLab workloads*

Chapter 6

Conclusions

Cloud computing made utility computing common nowadays. With such a vital resource it is reasonable to think that data centers will continue to proliferate in the future and accumulate a large fraction of the world's computing resources. For this reason, energy-efficient management of data center resources is an important problem for environmental (CO_2 emissions) and financial reasons.

This work has proposed three algorithms for implementing VMs consolidation. The proposed algorithms improve utilization while reducing the power consumption. A forecast made by Navigant Research indicates that research as the presented could facilitate the reduction of data center power expenditures from \$23.3 billion in 2010 to \$16.0 billion in 2020, as well as a 28% reduction in greenhouse gas emissions from the 2010 levels by adopting the Cloud computing paradigm for IT services delivery. Moreover, during the United Nations Climate Summit, the European Union announced their ambitious plans to reduce emissions as 40% by 2030, by being target of a broader objective of cutting emissions by 80-95% by 2050 U.N. Climate Summit 2014 .

To address the formulated research topics, this work has achieved each of the objectives presented in Chapter 1. In particular, in Chapter 2 a review of the concepts used on this work has been made. An overview on server virtualization, Cloud Computing, Service Level Agreement, an introduction to energy-efficiency on computing systems as well as the power models from the SPECpower were introduced. Finally an in-depth review of the optimization techniques used on this work was presented. Chapter 3 presented a review of the current state of the art on VMs consolidation techniques addressing power-efficient VMs placements.

Chapter 4 has proposed several approaches to consolidate VMs to achieve a power-efficient placement. The first is based on the knapsack problem, the second is based on a Evolutionary Computation approach, while the third considers doubling the virtual memory. Later those algorithm were modified to consider the use of virtual memory and also heuristics for linear and non-linear power models were proposed. An in-depth analysis of the proposed algorithms has been presented in Chapter 5. The experiments have been performed with two different traces, first using the PlanetLab project traces and later using the Google Cluster Data project traces.

Apart from the theoretical exploration, the presented work resulted in an open source implementation of pyCloudSim, a software framework for simulation of VMs consolidation in IaaS Clouds.

6.1 Future Research Directions

There are other methods that are likely candidates to solve the problem of VMs placement that would be interesting to explore, among them: Particle Swarm Optimization, Ant Colonies Optimization, Simulated Annealing, Tabu Search or Multi-commodity flow.

A specific approach has been proposed for the generation and evaluation functions on the Iterated-EC algorithm. There might be other alternatives that could even outperform the proposed ones. Also it would be possible to fine tune the parameters of the Iterated-EC algorithm to optimize the consumed time on computations. Another proposal is to run in parallel the algorithms, especially the Iterated-EC algorithm, due to the parallel nature of genetic algorithms.

Regarding the simulation framework, multi-threading improvements can be done to the simulator so simulations can be delegated to different cores. It is important to only delegate full simulation scenarios which are thread-save.

It worth researching about making decisions on the distribution of resources among the physical hosts based on policies, for example, it would be possible to have a physical servers pool for CPU intensive VMs, another one for I/O intensive, etc. Another approach would be the opposite, try to distribute equally the resources fairly across the physical hosts to optimize their use.

Even when pyCloudSim supports to specify a SPECpower model for each server individually, the experiments were performed with the same SPECpower model for all the servers, either linear or non-linear heuristics depending on the case. It would be possible possible to adapt the algorithms to consider heterogeneous data centers. Furthermore, it might be possible to create an adaptive non-linear heuristic to try to determine a global power efficient optimum placement.

Including dynamic behavior into the VMs and take Service Level Agreements into account it is an important feature in efficiently allocation VMs. It would be possible to even divide the processing of the solution to the servers of the cloud themselves. Likewise it would be possible to forecast and estimate economic savings during a period of time.

6.2 Final Remarks

Cloud computing has dramatically changed the landscape of application hosting in science, business, social networking as many other areas. Energy-efficient management of the resources within the Cloud infrastructure will enable Cloud providers to offer a scalable service while lowering the power requirements and CO_2 emissions. Research such as presented on this work could help with innovation in Cloud computing while help to preserve the environment.

Bibliography

Al Shayeji e Samrajesh (2012) M.H. Al Shayeji e M.D. Samrajesh. An energy-aware virtual machine migration algorithm. Em *2012 International Conference on Advances in Computing and Communications (ICACC)*, páginas 242–246. doi: 10.1109/ICACC.2012.55. Cited on page 24

AMD-V () AMD-V. AMD-V, 2015. URL http://www.amd.com/en-us/solutions/servers/virtualization. Accessed: 2015-01-18. Cited on page 5, 6

Barroso e Holzle (2007) L.A. Barroso e U. Holzle. The case for energy-proportional computing. *Computer*, 40(12):33–37. ISSN 0018-9162. doi: 10.1109/MC.2007.443. bibtex: 4404806. Cited on page 10, 45

Bazaraa *et al.* (2009) Mokhtar S. Bazaraa, John J. Jarvis e Hanif D. Sherali. *Linear Programming and Network Flows*. Wiley, Hoboken, N.J, 4 edition ed. ISBN 9780470462720. Cited on page 14, 16

Beloglazov e Buyya (2013) A. Beloglazov e R. Buyya. Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *IEEE Transactions on Parallel and Distributed Systems*, 24(7):1366–1379. ISSN 1045-9219. doi: 10.1109/TPDS.2012.240. Cited on page 2, 22

Beloglazov e Buyya (2012a) Anton Beloglazov e Rajkumar Buyya. OpenStack meat: A framework for dynamic consolidation of virtual machines in OpenStack clouds–a blueprint. Relatório técnico, Technical Report CLOUDS-TR-2012-4, Cloud Computing and Distributed Systems Laboratory, The University of Melbourne. URL http://www.cloudbus.org/reports/OpenStack-neat-Blueprint-Aug2012.pdf. Accessed: 2013-11-13. Cited on page 13

Beloglazov e Buyya (2012b) Anton Beloglazov e Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, 24(13):1397–1420. ISSN 1532-0634. doi: 10.1002/cpe.1867. URL http://onlinelibrary.wiley.com/doi/10.1002/cpe.1867/abstract. Cited on page 2, 21

Beloglazov *et al.* (2010) Anton Beloglazov, Rajkumar Buyya, Young Choon Lee e Albert Zomaya. A taxonomy and survey of energy-efficient data centers and cloud computing systems. arXiv e-print 1007.0066. URL http://arxiv.org/abs/1007.0066. bibtex: Beloglazov_ATaxonomyandSurveyofEnergyEfficientDataCentersandCloudComputingSystems_2010. Cited on page 1, 5, 6, 9, 10

BLAS (Basic Linear Algebra Subprograms) () BLAS (Basic Linear Algebra Subprograms). BLAS (Basic Linear Algebra Subprograms), 2015. URL http://www.netlib.org/blas/. Accessed: 2015-03-16. Cited on page 34

Brian *et al.* (2008) HAYES Brian, Thomas Brunschwiler, Heinz Dill, Hanspeter Christ, Babak Falsafi, Markus Fischer, Stella Gatziu Grivas, Claudio Giovanoli, Roger Eric Gisi e Reto Gutmann. Cloud computing. *Communications of the ACM*, 51(7):9–11. URL https://www.satw.ethz.ch/organisation/tpf/tpf_ict/box_feeder/2012-11-06_2_SATW_White_Paper_Cloud_Computing_EN.pdf. Cited on page 8

Calheiros *et al.* (2011) Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose e Rajkumar Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50. ISSN 1097-024X. doi: 10.1002/spe.995. URL http://onlinelibrary.wiley.com/doi/10.1002/spe.995/abstract. Cited on page 24

Chen *et al.* (2011) Ming Chen, Hui Zhang, Ya-Yunn Su, Xiaorui Wang, Guofei Jiang e K. Yoshihira. Effective VM sizing in virtualized data centers. Em *2011 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, páginas 594–601. doi: 10.1109/INM.2011.5990564. bibtex: 5990564. Cited on page 2

CloudAnalyst () CloudAnalyst. CloudAnalyst, 2014. URL http://cloudbus.org/students/MEDC_Project_Report_Bhathiya_318282.pdf. Accessed: 2014-01-17. Cited on page 24

CoMon - A Monitoring Infrastructure for Planetlab () CoMon - A Monitoring Infrastructure for Planetlab. CoMon - A Monitoring Infrastructure for Planetlab, 2015. URL http://comon.cs.princeton.edu/. Accessed: 2015-03-15. Cited on page 35

Creasy (1981) R.J. Creasy. The origin of the VM/370 time-sharing system. *IBM Journal of Research and Development*, 25(5):483–490. ISSN 0018-8646. doi: 10.1147/rd.255.0483. Cited on page 5

De La Fuente Vigliotti e Batista (2014) Albert De La Fuente Vigliotti e Daniel M. Batista. A green Network-Aware VMs placement mechanism. Em *Globecom 2014 - Symposium on Selected Areas in Communications: GC14 SAC Green Communication Systems and Networks (GC14 SAC Green Communication Systems and Networks)*, Austin, USA. Cited on page 3

Dhiman *et al.* (2010) G. Dhiman, K. Mihic e T. Rosing. A system for online power prediction in virtualized environments using gaussian mixture models. Em *2010 47th ACM/IEEE Design Automation Conference (DAC)*, páginas 807–812. bibtex: 5523620. Cited on page 2, 27

Fan *et al.* (2007) Xiaobo Fan, Wolf-Dietrich Weber e Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. Em *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, páginas 13–23, New York, NY, USA. ACM. ISBN 978-1-59593-706-3. doi: 10.1145/1250662.1250665. URL http://doi.acm.org/10.1145/1250662.1250665. bibtex: Fan:2007:PPW:1250662.1250665. Cited on page 10, 12, 22, 27, 35

Fang *et al.* (2013) Weiwei Fang, Xiangmin Liang, Shengxin Li, Luca Chiaraviglio e Naixue Xiong. VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers. *Computer Networks*, 57(1):179–196. ISSN 1389-1286. doi: 10.1016/j.comnet.2012.09.008. URL http://www.sciencedirect.com/science/article/pii/S1389128612003301. Cited on page 22

Feller *et al.* (2011) Eugen Feller, Louis Rilling e Christine Morin. Energy-aware ant colony based workload placement in clouds. Em *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*, GRID '11, páginas 26–33, Washington, DC, USA. IEEE Computer Society. ISBN 978-0-7695-4572-1. doi: 10.1109/Grid.2011.13. URL http://dx.doi.org/10.1109/Grid.2011.13. bibtex: Feller:2011:EAC:2082076.2082086. Cited on page 23

Feng (2014) Wu-chun Feng. *The Green Computing Book: Tackling Energy Efficiency at Large Scale*. CRC Press. ISBN 9781439819876. Cited on page 9

Fettweis e Zimmermann (2008) Gerhard Fettweis e Ernesto Zimmermann. ICT energy consumption-trends and challenges. Em *Proceedings of the 11th International Symposium on Wireless Personal Multimedia Communications*, volume 2, página 6. URL https://mns.ifn.et.tu-dresden.de/Lists/nPublications/Attachments/559/Fettweis_G_WPMC_08.pdf. Cited on page 1

Fraser (1960) Alex S Fraser. Simulation of genetic systems by automatic digital computers vi. epistasis. *Australian Journal of Biological Sciences*, 13(2):150–162. Cited on page 17

FuncDesigner () FuncDesigner. FuncDesigner, 2015. URL http://openopt.org/FuncDesigner. Accessed: 2015-03-16. Cited on page 33

Garey e Johnson (1978) M. R. Garey e D. S. Johnson. " strong " NP-completeness results: Motivation, examples, and implications. *J. ACM*, 25(3):499–508. ISSN 0004-5411. doi: 10.1145/322077.322090. URL http://doi.acm.org/10.1145/322077.322090. Cited on page 16

Garrett (2013) Aaron Garrett. Inspyred inspired intelligence initiative, 2013. URL http://inspyred.github.com. Accessed: 2013-11-13. Cited on page 29, 33

Gartner, Inc. () Gartner, Inc. Gartner estimates ICT industry accounts for 2 percent of global CO2 emissions, 2007. URL http://www.gartner.com/it/page.jsp?id=503867. Accessed: 2014-06-13. Cited on page 9

Goldberg (1989) David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, Reading, Mass, 1 edition ed. ISBN 9780201157673. Cited on page 17

Google Cluster Data Traces () Google Cluster Data Traces. Google Cluster Data Traces, 2015. URL http://code.google.com/p/googleclusterdata/. Accessed: 2015-03-15. Cited on page 34, 36

GreenCloud () GreenCloud. GreenCloud, 2014. URL http://greencloud.gforge.uni.lu/. Accessed: 2014-01-17. Cited on page 24

GridSim () GridSim. GridSim, 2015. URL http://sourceforge.net/projects/gridsim. Accessed: 2014-01-22. Cited on page 2

Holland (1975) John H Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press. Cited on page 17

Huppler *et al.* (2012) Karl Huppler, Klaus-Dieter Lange e John Beckett. SPEC: Enabling efficiency measurement. Em *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*, ICPE '12, páginas 257–258, New York, NY, USA. ACM. ISBN 978-1-4503-1202-8. doi: 10.1145/2188286.2188331. URL http://doi.acm.org/10.1145/2188286.2188331. bibtex: Huppler2012. Cited on page 11

IBM Announces Universal Management - Wake on LAN () IBM Announces Universal Management - Wake on LAN. IBM Announces Universal Management - Wake on LAN, 1998. URL https://www-03.ibm.com/press/us/en/pressrelease/2705.wss. Accessed: 2015-01-19. Cited on page 13

iCanCloud () iCanCloud. iCanCloud, 2014. URL http://www.arcos.inf.uc3m.es/~icancloud/Home.html. Accessed: 2014-01-17. Cited on page 24

Intel-VT () Intel-VT. Intel-VT, 2015. URL http://www.intel.com/content/www/us/en/virtualization/virtualization-technology/intel-virtualization-technology.html. Accessed: 2015-01-18. Cited on page 5, 6

Jackson e Lameter (2013) Paul Jackson e Christoph Lameter. CGROUPS kernel website, 2013. URL https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt. Accessed: 2013-08-08. Cited on page 6

Jaiantilal *et al.* (2010) Abhishek Jaiantilal, Yifei Jiang e Shivakant Mishra. Modeling CPU energy consumption for energy efficient scheduling. Em *Proceedings of the 1st Workshop on Green Computing*, GCM '10, páginas 10–15, New York, NY, USA. ACM. ISBN 978-1-4503-0450-4. doi: 10.1145/1925013.1925015. URL http://doi.acm.org/10.1145/1925013.1925015. Cited on page 11

Jong e Jong (2002) Kenneth A. de De Jong e Kenneth A. De Jong. *Evolutionary Computation.* A Bradford Book, Cambridge, Mass, 1st edition ed. ISBN 9780262041942. Cited on page 33

Jung *et al.* (2010) Gueyoung Jung, M.A Hiltunen, K.R. Joshi, R.D. Schlichting e C. Pu. Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures. Em *2010 IEEE 30th International Conference on Distributed Computing Systems (ICDCS)*, páginas 62–73. doi: 10.1109/ICDCS.2010.88. Cited on page 13

Kernel Based Virtual Machine () Kernel Based Virtual Machine. KVM (for kernel-based virtual machine) - main page, 2013. URL http://www.linux-kvm.org/page/Main_Page. Accessed: 2013-09-04. Cited on page 5, 6

Koomey (2011) Jonathan Koomey. Growth in data center electricity use 2005 to 2010. Relatório técnico, Oakland, CA: Analytics Press. URL http://www.analyticspress.com/datacenters.html. Cited on page 1

Koomey (2008) Jonathan G. Koomey. Worldwide electricity used in data centers. *Environmental Research Letters*, 3(3):034008. ISSN 1748-9326. doi: 10.1088/1748-9326/3/3/034008. URL http://iopscience.iop.org/1748-9326/3/3/034008. bibtex: Koomey2008. Cited on page 9

Kroshko (2013) Dmitrey L. Kroshko. OpenOpt website, 2013. URL http://openopt.org. Accessed: 2013-11-13. Cited on page 33

Lange *et al.* (2012) Klaus-Dieter Lange, David Schmidt, Andrew Bond e Lisa Roderick. SPECvirt_sc2010 - driving virtualization innovation. Em *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*, ICPE '12, páginas 251–252, New York, NY, USA. ACM. ISBN 978-1-4503-1202-8. doi: 10.1145/2188286.2188328. URL http://doi.acm.org/10.1145/2188286.2188328. Cited on page 11

LAPACK (Linear Algebra PACKage) () LAPACK (Linear Algebra PACKage). LAPACK (Linear Algebra PACKage), 2015. URL http://www.netlib.org/lapack/. Accessed: 2015-03-16. Cited on page 34

Linux-VServer Development Group () Linux-VServer Development Group. Linux-VServer, 2013. URL http://linux-vserver.org/Welcome_to_Linux-VServer.org. Accessed: 2013-09-05. Cited on page 6

Luke (2013) Sean Luke. *Essentials of Metaheuristics.* Lulu, second ed. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/. Cited on page 17, 20

LXC Development Group () LXC Development Group. LXC linux containers website, 2013. URL http://lxc.sourceforge.net/. Accessed: 2013-09-02. Cited on page 6

Martello e Toth (1990) Silvano Martello e Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations.* Wiley, Chichester ; New York, 1 edition ed. ISBN 9780471924203. Cited on page 16

Matplotlib () Matplotlib. Matplotlib, 2015. URL http://matplotlib.org/. Accessed: 2015-03-16. Cited on page 34

McGeer *et al.* (2010) R. McGeer, P. Mahadevan e S. Banerjee. On the complexity of power minimization schemes in data center networks. Em *2010 IEEE Global Telecommunications Conference (GLOBECOM 2010)*, páginas 1–5. doi: 10.1109/GLOCOM.2010.5683128. Cited on page 22

Mell e Grance (2011) Peter Mell e Timothy Grance. The NIST definition of cloud computing (draft). *NIST special publication*, 800(145):7. URL http://pre-developer.att.com/home/learn/enablingtechnologies/The_NIST_Definition_of_Cloud_Computing.pdf. Accessed: 2013-11-13. Cited on page 7, 8

Meng *et al.* (2010) Xiaoqiao Meng, Vasileios Pappas e Li Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. Em *2010 Proceedings IEEE INFOCOM*, páginas 1–9. doi: 10.1109/INFCOM.2010.5461930. bibtex: 5461930. Cited on page 2

Navigant Research () Navigant Research. Cloud Computing to Reduce Global Data Center Energy Expenditures by 38% in 2020, 2014. URL http://www.navigantresearch.com/newsroom/cloud-computing-to-reduce-global-data-center-energy-expenditures-by-38-in-2020. Accessed: 2014-12-10. Cited on page 67

NumPy () NumPy. NumPy, 2015. URL http://www.numpy.org/. Accessed: 2015-03-16. Cited on page 33

OpenStack Manual - Chapter 5 - Scaling () OpenStack Manual - Chapter 5 - Scaling. OpenStack Manual - Chapter 5 - Scaling, 2015. URL http://docs.openstack.org/openstack-ops/content/scaling.html. Accessed: 2015-01-16. Cited on page 1

OpenVZ project () OpenVZ project. OpenVZ linux containers wiki, 2013. URL http://openvz.org/Main_Page. Accessed: 2013-09-05. Cited on page 5, 6

Oracle VM VirtualBox () Oracle VM VirtualBox. VirtualBox, 2013. URL http://www.virtualbox.org/. Accessed: 2014-11-04. Cited on page 5

Planetlab Traces () Planetlab Traces. Planetlab Traces, 2015. URL https://github.com/vonpupp/planetlab-workload-traces. Accessed: 2015-03-15. Cited on page 34, 35

Rabinovich e Wigderson (1999) Yuri Rabinovich e Avi Wigderson. Techniques for bounding the convergence rate of genetic algorithms. *Random Structures & Algorithms*, 14(2): 111–138. ISSN 1098-2418. doi: 10.1002/(SICI)1098-2418(199903)14:2<111::AID-RSA1>3.0.CO;2-6. URL http://onlinelibrary.wiley.com/doi/10.1002/(SICI)1098-2418(199903)14:2<111::AID-RSA1>3.0.CO;2-6/abstract. Cited on page 20

Rajkumar Buyya (2013) Nikolay Grozev Rajkumar Buyya, Rodrigo N. Calheiros. CloudSim website, 2013. URL www.cloudbus.org/cloudsim. Accessed: 2013-12-04. Cited on page 2

Ranjana e Raja (2013) R. Ranjana e J. Raja. A survey on power aware virtual machine placement strategies in a cloud data center. Em *2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE)*, páginas 747–752. doi: 10.1109/ICGCE.2013.6823533. Cited on page 22, 23

Reiss *et al.* (2011) Charles Reiss, John Wilkes e Joseph L. Hellerstein. Google cluster-usage traces: format + schema. Technical report, Google Inc., Mountain View, CA, USA. Revised 2012.03.20. Posted at URL http://code.google.com/p/googleclusterdata/wiki/TraceVersion2. Cited on page 55, 56

Reiss *et al.* (2012) Charles Reiss, John Wilkes e Joseph L. Hellerstein. Obfuscatory obscanturism: making workload traces of commercially-sensitive systems safe to release. Em *3rd International Workshop on Cloud Management (CLOUDMAN)*, páginas 1279–1286, Maui, HI, USA. IEEE. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6212064. Cited on page 56

Sindelar *et al.* (2011) Michael Sindelar, Ramesh K. Sitaraman e Prashant Shenoy. Sharing-aware algorithms for virtual machine colocation. Em *Proceedings of the Twenty-third Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '11, páginas 367–378, New York, NY, USA. ACM. ISBN 978-1-4503-0743-7. doi: 10.1145/1989493.1989554. URL http://doi.acm.org/10.1145/1989493.1989554. Cited on page 16

Soltesz *et al.* (2007) Stephen Soltesz, Herbert Pötzl, Marc E. Fiuczynski, Andy Bavier e Larry Peterson. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. Em *ACM SIGOPS Operating Systems Review*, volume 41, páginas 275–287. URL http://dl.acm.org/citation.cfm?id=1273025. Cited on page 8

SPECPower Fujitsu PRIMERGY RX1330 Server () SPECPower Fujitsu PRIMERGY RX1330 Server. SPEC power_ssj2008 Benchmark of a Fujitsu PRIMERGY RX1330 M1 Server, 2007. URL https://www.spec.org/power_ssj2008/results/res2014q3/power_ssj2008-20140804-00662.html. Accessed: 2015-01-19. Cited on page xi, 13, 31, 40

SPECPower Sugon I840-G25 Server () SPECPower Sugon I840-G25 Server. SPEC power_ssj2008 Benchmark of a Sugon I840-G25 Server, 2007. URL https://www.spec.org/power_ssj2008/results/res2014q3/power_ssj2008-20140615-00658.html. Accessed: 2015-01-19. Cited on page xi, 31, 36

Standard Performance Evaluation Corporation (a) Standard Performance Evaluation Corporation. Standard Performance Evaluation Corporation, 2014a. URL http://www.spec.org/. Accessed: 2014-08-11. Cited on page 11, 21

Standard Performance Evaluation Corporation (b) Standard Performance Evaluation Corporation. SPEC power_ssj2008 Benchmark, 2007b. URL https://www.spec.org/power_ssj2008/. Accessed: 2015-01-19. Cited on page 11

Trygar e Bain (2005) T. Trygar e G. Bain. A framework for service level agreement management. Em *IEEE Military Communications Conference, 2005. MILCOM 2005*, páginas 331–337 Vol. 1. doi: 10.1109/MILCOM.2005.1605706. Cited on page 8

Uhlig *et al.* (2005) R. Uhlig, G. Neiger, D. Rodgers, A.L. Santoni, F.C.M. Martins, A.V. Anderson, S.M. Bennett, A. Kagi, F.H. Leung e L. Smith. Intel virtualization technology. *Computer*, 38(5): 48–56. ISSN 0018-9162. doi: 10.1109/MC.2005.163. Cited on page 6

U.N. Climate Summit 2014 () U.N. Climate Summit 2014. Speech by EU Commission President Barroso at UN Climate Summit 2014, 2014. URL http://eu-un.europa.eu/articles/fr/article_15491_fr.htm. Accessed: 2014-12-10. Cited on page 67

Verma *et al.* (2008) Akshat Verma, Puneet Ahuja e Anindya Neogi. pMapper: Power and migration cost aware application placement in virtualized systems. Em Valérie Issarny e Richard Schantz, editors, *Middleware 2008*, number 5346 in Lecture Notes in Computer Science, páginas 243–264. Springer Berlin Heidelberg. ISBN 978-3-540-89855-9, 978-3-540-89856-6. URL http://link.springer.com/chapter/10.1007/978-3-540-89856-6_13. Cited on page 23

Vigliotti e Batista (2014) Albert P.M.de la Fuente Vigliotti e Daniel Macedo Batista. Energy-efficient virtual machines placement. Em *2014 Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)*, páginas 1–8. doi: 10.1109/SBRC.2014.1. Cited on page 3

VMware Inc. () VMware Inc. VMware Inc., 2013. URL http://www.vmware.com/. Accessed: 2013-09-04. Cited on page 5, 6

Wei *et al.* (2011) Bing Wei, Chuang Lin e Xiangzhen Kong. Energy optimized modeling for live migration in virtual data center. Em *2011 International Conference on Computer Science and Network Technology (ICCSNT)*, volume 4, páginas 2311–2315. doi: 10.1109/ICCSNT.2011.6182436. Cited on page 23

Wu *et al.* **(2012)** Yongqiang Wu, Maolin Tang e W. Fraser. A simulated annealing algorithm for energy efficient virtual machine placement. Em *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, páginas 1245–1250. doi: 10.1109/ICSMC.2012.6377903. Cited on page 23

Xavier *et al.* **(2013)** M.G. Xavier, M.V. Neves, F.D. Rossi, T.C. Ferreto, T. Lange e C.A.F. De Rose. Performance evaluation of container-based virtualization for high performance computing environments. Em *2013 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, páginas 233–240. doi: 10.1109/PDP.2013.41. Cited on page 6

Xen Project () Xen Project. The xen project, the powerful open source industry standard for virtualization., 2013. URL http://www.xenproject.org/. Accessed: 2013-09-04. Cited on page 6

Xu e Fortes (2010) Jing Xu e J. A B Fortes. Multi-objective virtual machine placement in virtualized data center environments. Em *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, páginas 179–188. doi: 10.1109/GreenCom-CPSCom.2010.137. bibtex: Xu2010. Cited on page 22

Zelkowitz (2011) Marvin Zelkowitz. *Advances in Computers.* Academic Press. ISBN 9780123855138. bibtex: Zelkowitz2011. Cited on page 35