PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
FACULTY OF INFORMATICS
COMPUTER SCIENCE GRADUATE PROGRAM

PERFORMANCE-AWARE ENERGY-EFFICIENT CLOUD ORCHESTRATION

FÁBIO DINIZ ROSSI

Dissertation submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fullfillment of the requirements for the degree of Ph. D. in Computer Science.

Advisor: Prof. César Augusto Fonticielha De Rose

Porto Alegre
2016

Ficha Catalográfica

R831p Rossi, Fábio Diniz

Performance-Aware Energy-Efficient Cloud Orchestration / Fábio Diniz Rossi . – 2016.
114 f.
Tese (Doutorado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. César Augusto Fonticielha De Rose.

1. cloud. 2. energy-savings. 3. orchestration. 4. performance. 5. trade-off. I. De Rose, César Augusto Fonticielha. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE TESE DE DOUTORADO

Tese intitulada *"E-eco: Performance-Aware Energy-Efficient Cloud Orchestration"* apresentada por Fábio Diniz Rossi como parte dos requisitos para obtenção do grau de Doutor em Ciência da Computação, aprovada em 13 de maio de 2016 pela Comissão Examinadora:

Prof. Dr. César Augusto Fonticielha De Rose– Orientador — PPGCC/PUCRS

Prof. Dr. Tiago Coelho Ferreto – — PPGCC/PUCRS

Prof. Dr. Rodrigo Neves Calheiros - — The University of Melbourne

Prof. Dr. Daniel Gonçalves Gomes - — Universidade Federal do Ceará - UFC

Homologada em 06/10/2016, conforme Ata No. 020 pela Comissão Coordenadora.

Prof. Dr. Luiz Gustavo Leão Fernandes
Coordenador.

PUCRS PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Campus Central
Av. Ipiranga, 6681 – P. 32 – sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

I dedicate this work to my wife, Andréia.

“The real voyage of discovery consists not in seeking new lands but seeing with new eyes.” (Marcel Proust)

ACKNOWLEDGMENTS

Eu gostaria de agradecer a todos que me auxiliaram durante o doutorado e que contribuíram para a finalização desta tese. Em especial, eu gostaria de agradecer ao meu orientador, Prof. César A. F. De Rose, que me oportunizou a realização do doutorado, e com sua orientação ensinou-me os passos da pesquisa.

Agradeço aos membros da banca de defesa do doutorado: Prof. Rodrigo Neves Calheiros, Prof. Tiago Coelho Ferreto e Prof. Danielo Gonçalves Gomes pelos tempo investido e pelos comentários que vieram a reforçar o documento final. Aqui faço um agradecimento especial ao Prof. Rodrigo Neves Calheiros, que gentilmente me recebeu na Austrália durante o estágio sanduíche, e também ao Prof. Tiago Coelho Ferreto pela amizade e companheirismo durante o período na PUCRS.

Eu também gostaria de agradecer ao Prof. Rajkummar Buyya e aos membros do CLOUDS Lab da Universidade de Melbourne, que oportunizaram a discussão de meu trabalho, sempre na intenção de fortalecer minha pesquisa.

Eu não poderia deixar de agradecer aos colegas do LAD/PUCRS e PUCRS pelo apoio durante o doutorado, em especial ao Marcelo Veiga Neves, Miguel Gomes Xavier, Rafael Lorenzo Belle, Mauro Storch, Israel Campos de Oliveira, Raquel Dias, Kassiano José Matteussi, Timoteo Alberto Peters Lange, Endrigo D'Agostini Conte, Marcelo da Silva Conterato, Yuri Jaeger Monti, Guilherme Rocha dos Santos, Lucas Koinwaski Mior, Adalto Selau Sparremberger, Paolo Cemim, Luis Carlos Jersak, Carlos Roberto Moratelli, e Filipo Novo Mór (espero não ter esquecido ninguém).

Finalizando, eu gostaria de agradecer a minha esposa Andréia dos Santos Sachete, pelo apoio incondicional. Aos amigos Guilherme da Cunha Rodrigues, Luciele Gonçalves da Silva, e Arthur Francisco Lorenzon, agradeço pelo companheirismo e suporte, sempre que precisei.

ORQUESTRAÇÃO DE TÉCNICAS E MECANISMOS DE ECONOMIA DE ENERGIA CONSCIENTE DO DESEMPENHO DAS APLICAÇÕES EM NUVEM

RESUMO

O alto consumo de energia dos centros de dados tem sido foco na maioria das pesquisas recentes. Em ambientes de nuvem, várias soluções estão sendo propostas com o objetivo de alcançar eficiência energética, que vão desde o dimensionamento da frequência de processadores, da utilização de estados suspensão, até a consolidação de servidores virtuais. Embora estas soluções permitam redução no consumo de energia, apresentam impacto sobre o desempenho das aplicações. Visando resolver esta limitação, nós apresentamos uma orquestração de diferentes técnicas e mecanismos de economia de energia, com a finalidade de melhorar o balanceamento entre economia de energia e desempenho das aplicações. Para este fim, implementamos o e-eco, um sistema de gestão que atua juntamente com a plataforma de nuvem, decidindo qual estado de suspensão pode ser melhor aplicado sobre os servidores, além de escolher a melhor opção entre consolidação de servidores ou dimensionamento de frequência dos processadores. Para avaliar o e-eco, testes foram realizados em ambientes de nuvem real e simulado, utilizando aplicações scale-out em uma infraestrutura de nuvem dinâmica, levando em consideração transações-por-segundo como métrica de desempenho. Os resultados mostraram que nossa proposta apresenta a melhor relação entre economia de energia e desempenho de aplicações em ambientes de nuvem, quando comparada com outros trabalhos apresentados na literatura.

Palavras-Chave: balanceamento, desempenho, economia de energia, nuvem, orquestração.

PERFORMANCE-AWARE ENERGY-EFFICIENT CLOUD ORCHESTRATION

ABSTRACT

The high energy consumption of data centers has been a recurring issue in recent research. In cloud environments, several solutions are being used that aim for energy efficiency, ranging from scaling the processors frequency, through the use of sleep states, to virtual machine placement mechanism. Although these solutions enable the reduction in power consumption, they usually impact on the application performance. To address this limitation, we present an orchestration of different energy-savings techniques and mechanisms to improve the trade-off between energy savings and application performance. To this end, we implemented the Energy-Efficient Cloud Orchestrator – e-eco – a management system that acts along with the cloud platform, deciding which sleep state can be better applied on hosts, in addition to choosing the best option between virtual machines consolidation and dynamic processor frequency scaling. To evaluate e-eco, tests were carried out in a real and a simulated environment using scale-out applications on a dynamic cloud infrastructure, taking into account transactions per second as a performance metric. Results showed that our proposal presents the best improvement on the trade-off between energy savings and performance applications for cloud environments when compared with other works presented in the literature.

Keywords: cloud, energy-savings, orchestration, performance, trade-off.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1. INTRODUCTION

Computing aims to automate processes and to offer solutions based on the digital processing. With the increase of tasks to be automated, there is a growing need for greater processing power. Therefore, in recent decades, the architecture of computers, namely the processor, was subject to a constant increase of its capacity. Such increase was shown by Moore's Law [MVM09] as a trend that in a period of 18-24 months, chips would approximately double their capacity regarding transistors density.

The increase in transistors density causes a problem known as gate leakage, which is the loss of energy to each of these semiconductor switching, requiring greater energy input for the correct operation of transistors. At the same time, the clock rate of processors also increased. However, to handle the increase in the processor clock rate, the voltage that powers the processor must also increase. Both issues have allowed the expansion of the capabilities of processors, but they also resulted in higher power consumption. With the miniaturization of components that have enabled the development of denser processors and the pursuit of faster clock rates, the physical silicon limits [Boh98] were reached, due to the impact of energy consumption on overheating. A processor with a high temperature can produce, in a short term, failures in computer systems and, in a long time, a significant reduction of this component lifetime. Therefore, when these limits were achieved, it was not economically viable to develop faster processors. Two possibilities to overcome this limitation became reality.

The first solution was to replicate processors units. Thus, multi-core processors [RFAA13] came to support parallel applications that divide the workload across all available processors units, again to increase application performance. Thus, each core does not need to work at such a high frequency.

The second solution was supported by the development of other computing components such as network speed and memory size. Distributed architectures relying on large amounts of hosts arose to support high-performance processing and high availability: computing clusters [Buy99]. High-performance clusters have been widely adopted by academia and industry, which allowed increasing applications' performance due to the intensive use of resources. To enable greater processing capacity, clusters increased the quantity and replication of specialized components.

Each solution presents a set of advantages and disadvantages. On the one hand, parallel programming is less complex when compared to distributed programming, but it impacts the limit of available cores on the host, in addition to the competition for resources [BAK99]. On the other hand, distributed programming allows greater scalability, but its programming is more complex [MG11] [FGJN+10] [MKD04].

The multi-core technology has become a standard in modern computers, and it was adopted in most of the current clusters, adding the capabilities of parallel processing to distributed environments. This processing capability boosted the development of other components such as memory and network, providing more and more the increase in application performance.

However, with the increase of resources and their use by applications, related operating costs have also increased. Among these costs, power consumption [ABB+11] is one of the most important. Due to this fact, the U.S. Department of Energy's (DoE) Exascale Initiative Steering Committee proposed an initiative with the intention of limiting the power consumption for the new generation of exascale supercomputers in 20MW [SDM11]. This value has been set up based on the high cost per megawatt per year to operate data centers, including aspects as cooling and power conversion. DoE believed that this is the maximum threshold that can still keep the total cost of ownership of supercomputers manageable. Besides the financial aspect, the high consumption of energy impacts on sustainability issues such as emission of heat and harmful gasses to the environment [AN11].

When the energy consumption is again a limiting aspect for the growth of processing capacity combined with some factors, such as the advent of virtualization technology [SML10] and the concept of service-oriented architectures [HPD+14], a new paradigm drove the growth of a new large-scale computing environment: the cloud.

Virtualization is basically the provision of virtual infrastructure on a physical structure. The term can also be understood as a technology that enables to run multiple operating systems and applications simultaneously on the same machine. In this way, virtualization allows better use of available resources through elasticity. Service-Oriented Architecture (SOA) is a software design approach that promotes the integration between business and IT through well-defined service interfaces to provide the functionality of the applications in standard and interrelated services. SOA is composed of application components, grouped into a system to fulfill a precise function in the context of a well-defined architecture, providing well-defined interfaces that allow the connection and the use of their resources. In this way, SOA allows new applications to be created from the combination of components, called services. These services can be based on new developments or can be created from existing application functionality and, once created, a service can be reused and recombined to compose different systems.

As the load imposed by services' customers fluctuates, the infrastructure that supports such services must be specialized enough to meet those frequent changes. This is the ideal scenario for the use of cloud computing, which refers essentially to the idea of using, anywhere and platform the most varied applications through the Internet. Also, the pay-per-use model offered by cloud environments can reduce company costs. This model can only be supported by the most important characteristic of cloud environments, elasticity.

As a consequence of more effective use of resources through elasticity [GB12]—the application's ability to adjust the amount of resources needed to meet customers' requests while maintaining the same level of service quality—in addition to on-demand services, the cloud model offers as a computational environment where data centers can meet customer's needs, reducing costs for both cloud provider and their clients (users). Therefore, more than a technological breakthrough, cloud environments promote an economic paradigm that, through efficient management of resources, can reduce operating costs. Among these costs, currently one of the most important is the energy consumption of large data centers. Again, not only are the financial costs for in energy consumed, but all other expenses resulting from this consumption, such as heat dissipation and carbon dioxide emissions.

These savings can occur due to the intrinsic characteristics of cloud applications. Cloud data centers are large, and the behavior of applications is only intensive seasonally. A classic example is a video streaming service such as Netflix [DK13b], which throughout the week has a low or average usage rate, and on weekends or premiere dates, the number of users can increase quickly. Thus, the cloud environment keeps a certain minimum amount of machines to support the requests of customers, and during peak times, more resources are added to the pool resources to ensure the same quality of service. The MapReduce model [Mal11] is another example of this variable use of resources, wherein Map and Reduce phases generally do not use all amount of hosts at the same time.

Most of the time, this behavior leads to a significant number of idle computers. Although a large number of works propose ways of using the elasticity to save energy based on movement and location of virtual machines [CFF14], there are still energy-saving opportunities that can be utilized in a complementary way. Aiming to take advantage of this scenario, several studies have proposed strategies managing energy-saving techniques and mechanisms supported by new hardware technologies and operating systems. Such techniques and mechanisms range from the reduction of the processors frequency [KM08] and smart data routing [JS10], to sleep states replacing the idle hosts state to other deeper energy-savings states [GAL14].

Based on above, the main idea of this work is to take advantage of excess power consumed by idle hosts, through the intelligent use of such techniques and mechanisms, to save energy without burdening application performance.

1.1 Motivation

Information technology (IT) is increasingly present in the processes of modern society. Due to the needs of customers of IT services, priorities have always been focused on aspects of performance such as processing speed, bandwidth, memory, and storage capac-

ity. However, breakthroughs and greater use of available computing resources supported this increase in performance. The increase in the use of resources causes an environmental impact that had been underestimated until a few years ago when climate change and environmental pollution problems have become a priority for most nations.

The more energy is consumed by the data center, the greater the heat dissipated, and the greater the need for cooling. However, cooling systems are also impacting on a lot of electricity consumption. Therefore, power consumption has become environmental and economic issues. Furthermore, the increased processing power has only been feasible with the increase in energy consumption. It seems logical that somehow energy savings necessarily imply in decreased performance. In the same way, Cameron [Cam14] presents two essential relevant discoveries: (1) substantial performance loss can occur when power management is enabled, and (2) for the data points measured when performance loss occurred, the standard deviation in many cases exceeded the mean.

Still, Cameron [Cam14] argues that the performance loss is recurrent, and finds out the cause requires a great effort. Besides, the author believes that data center operators can lose their jobs due to a substantial performance loss at a critical time. Thus, data center operators prefer to disable power management so that these features do not interfere with performance. In short, power management does not result in job risks to operators. However, this is not true when poor performance management happens.

Thus, the reduction of energy consumption is a challenge for the provider of cloud services, as they must support the increased demands and maintain the performance expected by customers. By themselves, the techniques and mechanisms provide energy savings, but at a high cost on performance by reducing the processor frequency. Such a mechanism reduces the amount of instructions that the processor can execute at a given time interval, increasing the process runtime. The action of turning off hosts, understood as a great opportunity to save energy, impacts on application performance when an environment has an organic behavior, with many context switches in small time intervals. The time during the transitions between turn on and off hosts should be added to the total time of process execution. Even migration of virtual machines provided by virtualization must stop the services running on virtual machines for some time, enabling the copy of the application context from a physical machine to another. This raises the need for strategies to improve the energy efficiency of cloud computing data centers, and at the same time to preserve desired levels of operation.

The motivation of this work starts from the assumption that it is possible to improve the trade-off between energy savings and application performance in cloud environments, taking into consideration the overhead imposed by each technique and mechanism on performance metrics. The knowledge about these overheads provides choices on which energy saving techniques and mechanisms can suit better the cloud applications behavior. These options enable data center operators to be confident that the application of energy saving

in the data center will not incur negative impact in the applications, and therefore motivates operators to utilize them, contributing to increasing the adoption of such techniques.

Objective

Based on the above observations, the management of power saving techniques incurring smaller impact on applications performance introduces a new challenge. This dissertation aims to improve the trade-off between energy saving and applications performance, through an on-the-fly orchestration of energy techniques and mechanisms coupled with the cloud application behavior, reducing the impact of these techniques and mechanisms on application performance. To this end, this work identified two situations where these techniques and mechanisms can be managed. Firstly, based on performance metrics provided by applications running on active hosts, it decides a number of hosts to be maintained in the standby state to meet a possible future demand, keeping the rest of the hosts turned off. Also, based on the organization of virtual machines, it automatically chooses either reducing the processors frequency or performing virtual machines consolidation along with turning off operation for idle hosts.

1.2 Hypothesis and Research Questions

The main goal of this Ph.D. research is to investigate the hypothesis that an effective orchestration of several energy-saving techniques and mechanisms on cloud environment can save power with a minimum impact on performance metrics.

To guide this investigation, fundamental research questions associated with the hypothesis are defined as follows:

1. *Where are energy saving techniques and mechanisms available for cloud environments derived from and how is its management carried out?* This research question's main objective is to investigate the techniques and mechanisms offered by hardware technologies and operating systems present in today's data center aimed at energy savings. It is important to determine the way these hardware and software techniques and mechanisms can be managed, aiming to develop a strategy based on the impact that each one causes of application performance metrics.

2. *What is the impact of energy-saving techniques and mechanisms on performance metrics in a single computer host?* The objective of this research question is to verify how energy saving techniques and mechanisms impact on performance metrics in a computer host. This is important to check alternatives to improve the trade-off between applications performance and energy savings.

3. *What is the impact of energy-saving techniques and mechanisms on performance metrics on a cluster?* The motivation for this research question comes from the decision on the impact of the network on energy-saving decisions and how those decisions affect the applications performance in such environments. Unlike evaluations conducted on only one host, assessments on environments interconnected by the network are important since they must take into account the network bandwidth concerning the data traffic during virtual machine migration.

4. *What is the difference between assessments previously achieved in this work (for one host e for cluster) compared to a cloud computing environment?* Once we understand the trade-off between performance and energy savings on a host and a cluster, it is necessary to decide what are the differences of these environments to cloud environments. This is important because of the different behavior of the services offered by the cloud, and the way energy saving techniques and mechanisms should be adapted to this behavior in order not to affect application performance.

5. *Is it possible to improve the trade-off between performance and energy saving in cloud data center environments?* All previous questions enable the understanding of the overheads imposed by the energy saving techniques and mechanisms. The results conduct the orchestration of the energy-saving techniques and mechanisms on cloud environments providing the least impact on application performance. To answer this question, we proposed the Energy-Efficient Cloud Orchestrator (e-eco), an orchestrator of energy-saving techniques and mechanisms on a cloud environment that aims to improve the trade-off between power savings and applications performance. It allows on-the-fly management of which techniques and mechanisms should be applied to the behavior of the application, reducing the impact of such techniques and mechanisms on application performance.

1.3 Methodology

The evaluation of the proposed strategy is carried out through two main methodologies:

1. *Prototyping:* Firstly, evaluations were performed on isolated hosts and set of hosts, enabling the understanding of each power saving technique and mechanism, and its influence on performance for each of these scenarios separately. Later, an orchestrator has been implemented in a real cloud infrastructure, which has shown its behavior on a small scale.

2. *Discrete-event simulation:* Apart from real implementation, simulation has been used enabling to expand the size of the infrastructure, to validate the orchestrator features

already verified for a small infrastructure. Thus, we can show that the results for the implementation of the orchestrator in a small cloud infrastructure can be replicated in large infrastructures.

Workload and Scenario

Applications offered as services on cloud environments have different characteristics from applications supported by other large-scale environments such as clusters and grids. While in more traditional environments the workload usually uses all the available resources, and it has a finite completion time, cloud applications are generally offered as continuous services indefinitely with variation amounting to resources required at the time. Therefore, due to the nature of cloud applications and the behavior of the requests performed by customers of this service, there is a fluctuation rate in the resources usage in such environment.

To support this fluctuation and aiming to maintain the quality of service within acceptable levels for the customer, the cloud infrastructure must dynamically increase the number of resources when necessary, and reduce resources when there is underutilization. This dynamicity is supported by the elasticity of the virtualization layer, which provides an optimal setting for applications while maintaining a low utilization rate most of the time, but which might quickly require a larger set of resources supporting new customer requests. Figure 1.1 exemplifies this behavior, in which customers (a) access a cloud service through a listener (b). When there is a need for more resources, replicas of the service are instantiated through a load balancer (c) on new available physical resources (d). When resources are no longer needed, the instances can be released, and the now idle resources are managed, aiming to save energy.

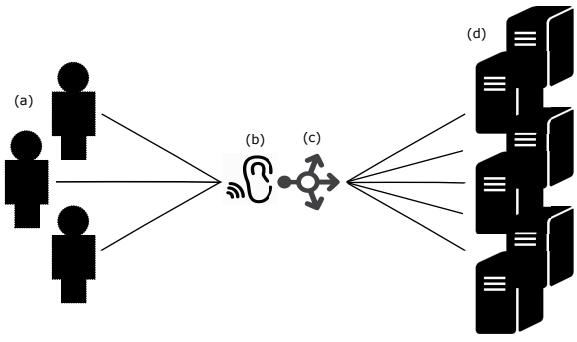


Figure 1.1 – Horizontal scalability in cloud infrastructures. Resources increase or decrease by adding or removing capacity in agreement with the policies and predefined metrics. This feature should support the demands of customers paying only for what they need and use. This elastic scenario enables the utilization of different power saving techniques, as explored in this dissertation.

Therefore, applications that perform elasticity on the resources for performance purposes represent an ideal setting for the implementation of this proposal. Another nec-

essary condition for the implementation of this proposal is a cloud environment where the service provider can manage the technologies in the infrastructure layer to modify it without any restrictions. For this reason, the dissertation proposed idea is suitable for private cloud environments, in which the service provider has full control and knowledge of underlying technologies.

1.4 Contribution

The main contributions of this dissertation can be broadly divided into 3 major categories: 1) cloud components terminology, literature review, and related work analysis, 2) the impact of energy saving techniques and mechanisms on performance at different levels of processing within a cloud architecture, 3) an orchestrator of energy-saving techniques and mechanisms in the cloud with concern the application performance and expansion of scenarios using simulation. The key contributions of the dissertation are as follows:

1. *Cloud components terminology, literature review, and related work analysis:* Terminology of energy savings solutions managed by cloud environments infrastructure platforms.
    - *Terminology:* a comprehensive bottom-up organization that disposes of, according to the level of abstraction, aspects of the infrastructure platform, from the hardware components to more abstract levels.
    - *Survey:* state-of-the-art literature review organization and discussion.
2. *The impact of energy saving techniques and mechanisms on performance at different levels of processing within a cloud architecture:* Studies showing the trade-off between several energy-saving techniques and mechanisms and application performance at different levels.
    - *On a host:* analyses of the opportunities for energy savings in a computational host, with emphasis on reducing the frequency of processors and their relationship with application performance.
    - *On a cluster:* expansion of previous evaluations for a set of hosts interconnected by a network. In addition to the reduction in the frequency of processors, sleep states are now considered as an option, and their impact on performance is evaluated.
    - *On cloud environments:* virtualization in cloud environments enable energy savings through the migration of virtual machines. Therefore, all the techniques and

mechanisms previously studied are coupled with this new opportunity to save energy. At this point, the analysis of the impact of these energy-saving proposals is focused on the behavior of services offered by the cloud.

- The choice about which techniques and mechanisms are best suited to the behavior of cloud services aimed at the smallest possible impact on application performance.
  - *Cloud services behavior:* discussion about the differences between "jobs" running on hosts and clusters, and "services" offered by cloud environments.
  - *Impact of energy-efficient techniques and mechanisms on cloud services:* analyses of the impact of energy saving techniques and mechanisms available in cloud environments on the performance of the services offered by the provider.

3. *An orchestrator of energy-saving techniques and mechanisms in the cloud with concern to the application performance and expansion of scenarios using simulation:* Implementation in real and simulated cloud infrastructures of a new orchestrator enabling balancing the cloud power-savings techniques and mechanisms with minimal impact on application performance.

   - *Energy-efficient cloud sites:* selection of energy-saving states that can provide more energy savings, while presenting the least impact on the performance of services, taking into account its time and energy consumption during transitions.
   - *Energy-efficient techniques and mechanisms selection on running hosts site:* decision either enabling the consolidation of virtual machines or reducing the frequency of processors, taking into account the location of virtual machine images.

1.5 Dissertation Organization

This dissertation aims to improve the trade-off between performance and power saving in cloud applications. Therefore, experiments on such trade-off on independent hosts and set of hosts (clusters) were performed. After, the differences between jobs in clusters and services in clouds were analyzed. The results showed the way to develop a final solution to improve such trade-off for cloud applications. This dissertation is organized showing in each chapter, each phase of such experiments.

Work presented in this dissertation has been partially or completely derived from a set of papers published during the Ph.D. candidature. Figure 1.2 shows the relationship among the chapters.

The remainder of this dissertation is organized as follows:

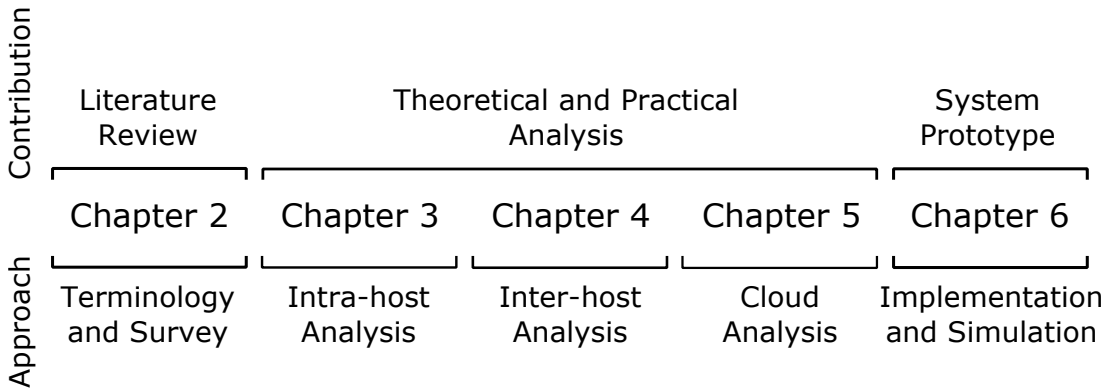


Figure 1.2 – Dissertation Organization.

- Chapter 2 presents the state-of-the-art concepts that are relevant for energy-aware studies. The chapter addresses research question 1 and it is partially derived from:
  - XAVIER, Miguel G.; **ROSSI, Fábio D.**; DE ROSE, César A. F.; CALHEIROS, Rodrigo N.; GOMES, Danielo G.. "Modeling and Simulation of Global and Sleep States in ACPI-compliant Energy-efficient Cloud Environments". *Concurrency and Computation: Practice and Experience (CCPE), Volume xx, Issue xx, Pages: xxxx-xxxx, John Wiley Sons, Ltd, USA, 201x*.
- Chapter 3 presents the relationship between energy-saving techniques presented in Chapter 2 and their impact on application performance in an intra-host environment. The chapter addresses research question 2 and it is partially derived from [RDR14a] [RDR15b]:
  - **ROSSI, Fábio D.**; XAVIER, Miguel G.; CONTE, Endrigo D.; FERRETO, Tiago; DE ROSE, César A. F.. "Green software development for multi-core architectures". *In: IEEE Symposium on Computers and Communication, 2014, ISCC 2014, Funchal, p. 1-6*.
  - **ROSSI, Fábio D.**; STORCH, Mauro; OLIVEIRA, Israel; DE ROSE, César A.F.. "Modeling Power Consumption for DVFS Policies". *In: IEEE International Symposium on Circuits Systems, 2015, ISCAS 2015, Lisbon. p. 1879-1882*.
- Chapter 4 expands the energy and performance evaluations carried out in Chapter 3 for a set of machines, adding to previous assessments, multiprocessing and communication issues. The chapter addresses research question 3 and it is partially derived from [RDR15c]:
  - **ROSSI, Fábio D.**; XAVIER, Miguel G.; MONTI, Yuri J.; DE ROSE, César A.F.. "On the Impact of Energy-Efficient Strategies in HPC Clusters". *In: 23rd Euromicro International Conference on Parallel, Distributed and Network Based Processing, 2015, PDP 2015, Turku, p. 17-21.*

- Chapter 5 introduces cloud environments compared to the interhost environment presented in Chapter 4. Further, it discusses which metrics should be adopted to evaluate this environment. The chapter addresses research question 4 and it is partially derived from [RDR14b]:

  – **ROSSI, Fábio D.**; CONTERATO, Marcelo ; FERRETO, Tiago ; DE ROSE, César A.F.. "Evaluating the Trade-off Between DVFS Energy-savings and Virtual Networks Performance". *In: The Thirteenth International Conference on Networks, 2014, ICN 2014, Nice, p. 274-279*.

- Chapter 6 describes the architecture and implementation of e-eco, an Energy-Efficient Cloud Orchestrator. In addition, it presents evaluations on real and simulated cloud environments. The chapter addresses research question 5 and it is partially derived from [RDR15a]:

  – **ROSSI, Fábio D.**; DE ROSE, César A.F.; CALHEIROS, Rodrigo N.; BUYYA, Rajkumar. "Non-Invasive Estimation of Cloud Applications Performance via Hypervisor's Operating Systems Counters". *In: The Fourteenth International Conference on Networks, 2015, ICN 2015, Barcelona, p. 177-184*

- Chapter 7 states the dissertation discussing conclusions and future research directions.

2. ENERGY-EFFICIENT CLOUD COMPUTING: FUNDAMENTAL CONCEPTS

Energy efficiency concerns achieving the best performance of a service with the least expenditure of energy. As an example of action, there is the modernization of equipment and processes to reduce their energy consumption. Another example is the application, with existing equipment and processes, of strategies to save energy with the least impact on performance.

To guide the understanding of this concept, this chapter presents a background on cloud computing, focusing on the elastic capacity provided by this environment. Afterward, it is introduced the ACPI standard, which enables the use of energy saving techniques in data centers. Based on the energy-saving techniques on proposed strategies, this chapter proposes a classification that allows for the organization of cloud environments infrastructure components as elements of the same domain. Finally, related work is described, addressing our proposal.

2.1 Cloud Computing

Cloud computing is widely spread in the industry of information technology (IT), becoming a standard infrastructure for offering IT services. Cloud computing is a paradigm that associates the service-oriented model with the flexibility provided by the characteristics of elastic infrastructure layer. Supported by these concepts, cloud computing can be defined as a model that enables access, on demand to a set of configurable computing resources (networks, servers, storage devices, applications, and services) that can be quickly acquired and released with a minimal management effort or service provider interaction. This means that cloud environments consist of a set of IT services accessed over the network, providing scalability, quality of service, and inexpensive computing infrastructure that can be accessed simply and pervasive way.

Cloud environments can be implemented to provide service quality levels and intelligent use of resources through service models. These service models define the architectural standard for the solutions offered in cloud environments, as seen in Figure 5.1. There are currently cloud models where SaaS executes directly on the IaaS or even SaaS and PaaS executing directly on the hardware. Thus, we present the characteristics of each of these three layers, without specifying a set of them as a standard for cloud environments.

In the software layer (Software-as-a-Service—SaaS), software for specific purposes is available, offered to customers through the Internet. The customer does not manage or control the underlying infrastructure (network, servers, operating systems, storage,

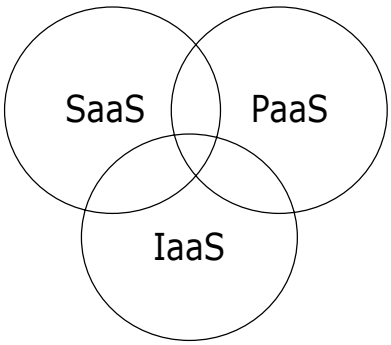


Figure 2.1 – Traditional Cloud Model

etc.). This approach enables developers to focus on innovation of services offered and not in infrastructure, to develop solutions in a more agile way. Because the software is accessible through the Internet, it can be accessed from anywhere and at any time, allowing greater integration between units of the same company or integration with other software services. This feature enables the incorporation of new features to services transparently to customers. As SaaS examples can be highlighted the SalesForce.com and IBM Lotus Live.

The platform layer (Platform-as-a-Service—PaaS) provides a high-level platform for integrating, implementing and testing cloud applications through APIs that abstract the operating system, and application stack (the web, application, and database servers). For the organization, PaaS provides management of work and responsibilities of information technology teams. As an example of PaaS, Google App Engine can be highlighted.

The main objective of the infrastructure layer (Infrastructure-as-a-Service—IaaS) is to provide computation resources (such as servers, network, storage, etc.) on demand, to meet customers requests within the levels of agreement contracted service. Requirements of services are met by infrastructure platforms that manage resources through various components such as load balancers and auto-scalers (dynamic change in the available number of resources on-demand). Thus, the infrastructure becomes scalable through the use of characteristics such as elasticity.

Although these three layers have been determined initially as a standard, cloud environments should not necessarily keep all of them, and some proposals include new layers among the three originally proposed.

Regarding deployment models, there are four types: private clouds, public clouds, hybrid clouds, and community clouds.

Private clouds are those in which a company has exclusive access to the resources of a cloud, whether it be their machine park or leased from a cloud provider. This model has the advantage of the exclusive use of resources, which increases control and data security in the cloud.

Public clouds, unlike private clouds, are offered to any individual or organization, which pay for the use of resources. Resources can be shared among all customers, and

generally, it is not controlled by the customer. The management is transparent to customers because the service provider assumes for itself the environmental maintenance responsibilities.

Hybrid clouds consist of private environments that at certain times of high demand, can allocate resources in a public cloud. This model assumes that the two cloud models, private and public, are connected by technologies that enable portability of applications.

Community clouds consist of infrastructures shared by a group of organizations that have common interests. The model can be managed by companies themselves or by third parties. Additionally, the infrastructure may be internal or external to the company.

In addition to the service models and implementation, some key features in cloud environments should be commented. Among them, the top five are: on-demand self-service, broad access, resource pooling, service measurement, and elasticity.

On-demand self-services enable customers to purchase computational resources, in so far as and without requiring the interaction with the service provider. This is supported due to the ability of cloud platforms to orchestrate resources in a dynamic way, and present these modifications to customers transparently. Broad access is allowed through cloud platforms that provide interfaces for different devices that can access services through the Internet, such as computers, mobile phones, tablets, among others. Pooling resources are the set of physical resources offered transparently to users. This transparency can abstract information about architectures and locations.

Measurement of service is also considered an important task in cloud environments because quality of service limits are marked by contracted service levels agreements by the customer to the provider, and this quality of service is what makes a number of necessary resources be offered. Therefore, the measurement metrics related to services and utilization rates of resources are decisive factors for making the cloud scalable, safe and available.

The last of the five characteristics, and one of the most important is the cloud's ability to be elastic. Elasticity means that depending on the demand created by applications, resources can be acquired or released at runtime. Therefore, it promotes, in a transparent way for customer, scalability driven by fluctuating workloads.

Elasticity

One of the main motivations for the use of cloud computing is its ability to acquire and release resources dynamically. From a customer point of view, the cloud seems to have infinite resources, as this can get more or fewer resources as the need for their applications. Liu et al. [LTM+12] defines elasticity as the ability of rapid over-provisioning and under-provisioning, with the capacity of virtual resources and virtually infinite amount obtainable without restriction and at any time. In turn, the service provider is responsible for the acquisition, management, and maintenance of the entire cloud infrastructure.

Commonly, the term elasticity is used synonymously with scalability. However, they are different concepts and should not be used as synonyms. Scalability is the system's ability to grow to a size to accommodate expected future demand or improve its performance, by adding new features. An application is deemed scalable if its efficacy is maintained as the amount of resources and the problem of size increases proportionately. Thus, scalability of an application reflects its ability to use available resources efficiently. Elasticity, on the other hand, consists of the proactive or reactive ability to increase or decreasing resources of a service at run time. As the elastic modalities in cloud environments, there is elasticity in two directions: vertical and horizontal.

Vertical elasticity (scale-up), usually consists of adding more processors or memory on the same host increasing its capacity. The advantages of this model consist of a lower energy consumption than applications running in a larger quantity of hosts, cooling costs are reduced due to the smaller amount of necessary hosts, less complex to implement, permits lower cost and uses a lower rate the network devices. On the other hand, there are single points of failure.

Horizontal Elasticity (scale-out) refers to the addition of more hosts to the environment. The advantage of this proposal is to a virtually infinite scalability although there are software limitations or other attributes of the infrastructure. As an advantage, scale-out provides load balancing and availability. Moreover, there is a significant emission of gasses at the data center, increase power and cooling costs, a greater quantity of network equipment is needed, and the applications that use these environments are more complex due to the required adaptability.

Server Consolidation

Both vertical and horizontal elasticity can save resources. In most cloud environments, the elasticity is provided by the virtualization layer, and this is performed by migration of virtual machines (VM) among the available physical resources.

VM migration moves a virtual machine from one physical machine to another, and this should be performed transparently to the guest operating system and VM remote customers. In times of low demand from customers, this feature allows VMs to be allocated in a smaller amount of resources, and this operation is known as server consolidation [Vog08].

As well as server consolidation saves resources, at the same time, this saves energy. This occurs because when VMs are migrated to a smaller amount of resources, the idle resources due to the VMs deallocation can support ACPI sleep states (e.g. turned off).

2.2 ACPI Architecture

ACPI is a specification that provides an open standard for operating system (OS) power management. It was designed to allow OSs to configure and control each hardware component, replacing both the predecessors Plug and Play (PnP) Energy Management and the Advanced Power Management (APM). In current hosts, the firmware-level ACPI is implemented in the ACPI BIOS code, which provides tables containing information on hardware access methods. OSs use this information for tasks like assigning interrupts or (de)activating hardware components. As the OS performs this management, there is greater flexibility regarding energy-saving modes for CPU and several other devices present in the hardware. This section outlines how ACPI is organized and how its components relate to each other.

The ACPI architecture can be seen in Figure 2.2, where the upper part represents user-mode applications and threads dispatched by OS. A device driver performs the communication between OS and hardware platform. Likewise, power management is carried out by the ACPI driver through communication between OS and the hardware platform.

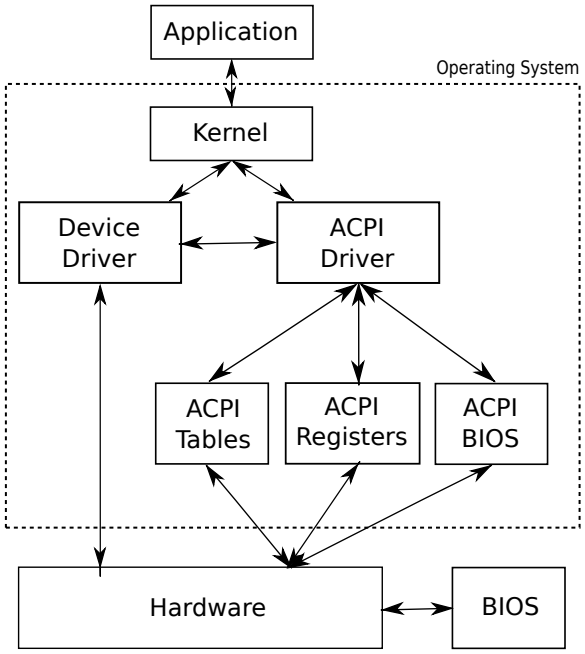


Figure 2.2 – The ACPI Architecture [Gro03]

The ACPI driver manages three different components: ACPI Tables, ACPI Registers, and ACPI BIOS. ACPI Tables contain hardware description managed through ACPI, including machine-independent bytecode used to perform hardware management operations. ACPI Registers provide low-level hardware management operations. Finally, during hardware designing, additional registers are implemented to be accessed through the bytecode stored in the device-specific part of the ACPI tables, referred to as ACPI BIOS.

In computers that support ACPI, before the OS is loaded, the ACPI BIOS puts the ACPI Tables in memory. Thus, when the OS is started, it searches for a small data structure with the valid signature within the BIOS and uses a pointer to the ACPI Tables to find the definition of hardware blocks. The ACPI Registers store changes that are made in the ACPI Tables. Through the ACPI, the OS can put devices in and out of low-power states. Devices that are in use can be turned off. Similarly, it uses information from applications and user settings to put the system as a whole into a low-power state.

Although ACPI is a standard used in today's computers, several legacy architectures remain in use in data centers. Most of these architectures have their resources managed directly by the firmware. The main firmware-based approach is the System Management Mode (SMM). As computer architectures evolved, conflicts started to be observed between information obtained via SMM and OS [DK13a]. Since in SMM the processor's states are stored in the System Management RAM (SMRAM), ACPI can read these legacy states and translate them to supported ACPI states.

2.2.1 ACPI States Overview

From a user-visible level, the system can be thought of as being in one of the power states presented in Figure 2.3. In the figure, the arrow indicates the depth of energy savings provided by each state. ACPI specifies different power state levels, which are: Global states, Sleep states, Device states, and Processor states. Some of these levels concern IT resources, such as computers, hard disks, graphic cards, in addition to other peripherals, such as the processor chip.

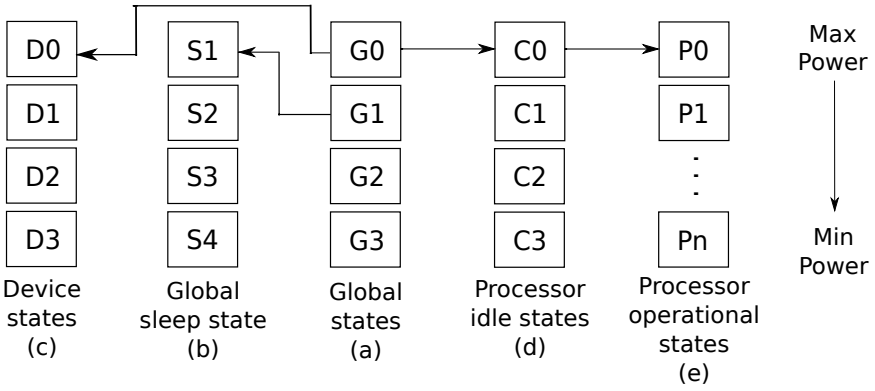


Figure 2.3 – ACPI states

Global states (Figure 2.3 (a)) denote the entire system and are visible only to the user. Sleep states (Figure 2.3 (b)) are power states derived from the G1 state and are visible only to the system. When the user presses the power button, for example. Power states of a particular device (Figure 2.3 (c)) are usually not visible to the user. For instance, devices

Table 2.1 – Description of the ACPI states

Global States	
G0	the system and user threads are running (working)
G1	the system consumes a small amount of power, user mode threads are not running, the system appears as if turned off, and the system context is saved (sleeping)
G2	the system consumes a minimal amount of power, user mode threads and system processes are not running, and the system context is not saved (Soft Off)
G3	the system is turned off (Mechanical Off)
Sleep States	
S1	no system context is lost
S2	CPU and system cache context are lost
S3	CPU, system cache, and chip set context are lost
S4	powered off all devices
Device States	
D0	device is turned on and running
D1	low-power state when the device context may or may not be lost
D2	low-power state when the device context may or may not be lost, and the power supply of the bus is reduced
D3	device is turned off and not running
Processor States	
C0	CPU is on
C1	CPU is not executing instructions
C2	CPU main internal clocks are stopped
C3	deep sleep
Processor Operational States	
P0	maximum processor performance capability and may consume maximum power
P1	the processor performance capability is limited below its maximum and consumes less than maximum power
Pn	the processor performance capability is at its minimum level and consumes minimal power while remaining in an active state

may be turned off while the system keeps working. Finally, Processor states (Figure 2.3 (d)) are power states within the G0 state (working state). It means the Processor states may vary if the computer is processing something. Besides those mentioned states, Dynamic Voltage and Frequency Scaling (DVFS) [KZS11] is the name given by the industry to P-states (Figure 2.3 (e)). Each level denotes one of all available modern CPU's frequencies, which in conjunction with ACPI-based firmware allows on-the-fly adjustment based on the CPU load. Table 2.1 shows the depth levels of the mentioned states, as well as their description. The

deeper the state, the lower the power rate and the higher the latency for returning to the working state.

2.2.2 Synthesis and Discussion

The first most widely used sleep state in energy-saving strategies is called standby (S1), which turns off the screen, hard drives, and fans. Because all running programs are kept stored in memory (either RAM or swap), the memory remains active, requiring little power for maintaining user data until some external event occurs and turns the subsystems back on. The advantage of this state is the short time needed for the computer to be on again. This is fundamental in situations where the computer must be awakened to all possible events or do so very quickly. As the context of the OS is stored in a volatile memory that requires power to keep up the data, there is a disadvantage when instabilities occur in the power grid.

Another lower-power state adopted in a variety of energy-saving strategies is called hibernate (S3). In this state, the computer is completely turned off, and the application execution context from the memory is stored as a file on the hard disk. When an external event interrupts hibernation, the computer is turned on, and the original state is loaded from the hard drive to the memory. Computers consume less power in this state because most of the hardware components are turned off. The drawback is that the computers in this state incur a higher latency for getting ready due to the cost of moving the context from disk to memory.

Besides these two power states, another way to save energy is by turning computers off without worrying about the OS state, application contexts, or user data. This behavior refers to the global state G2 in ACPI. The difference in this state compared to hibernation is that it does not keep settings in memory.

These states can be controlled locally by ACPI commands, but in some systems, the ACPI might also be remotely managed using Wake-on-LAN (WoL) [GMAMPMJGI06]. WoL consists of a standard developed by AMD for computers connected to a network to manage energy information. For this, the network card and the motherboard must support WoL.

2.3 Energy-Efficient IaaS Components Terminology

Most of the solutions for energy efficiency are located in the infrastructure layer of cloud environments. This is because this layer hosts the hardware and basic systems that support energy saving components. However, such solutions should guide the use of these

elements, with the intention of not causing interference on other metrics. Therefore, within the infrastructure, there are several abstraction levels of the same solution.

However, existing classification of cloud infrastructure components that enable energy-saving aspects is inconsistent. For example, the literature shows the Dynamic Voltage and Frequency Scaling (DVFS) sometimes classified as a technique [WCC14], a mechanism [SFBS+12], a policy [EE11], a strategy [LW14] or even a module [GMC+13]. In the same way, virtual machine (VM) migration is referred to as a technique [ILA+11], a mechanism [ZLOA13] or a strategy [FFCdM11].

Aiming to organize such different levels, we are proposing the bottom-up terms terminology presented in Figure 2.4, considering only the IaaS level. A platform software orchestrates components of the infrastructure layer. The infrastructure platform manages and interconnects different modules, each driven by a metric. Each of these modules includes a set of abstractions ranging from strategies to achieve the goal to the hardware support for this strategy to occur. Although the module described in Figure 2.4 refers to energy, the proposed terminology can be used similarly to each of the other modules belonging to an infrastructure platform.

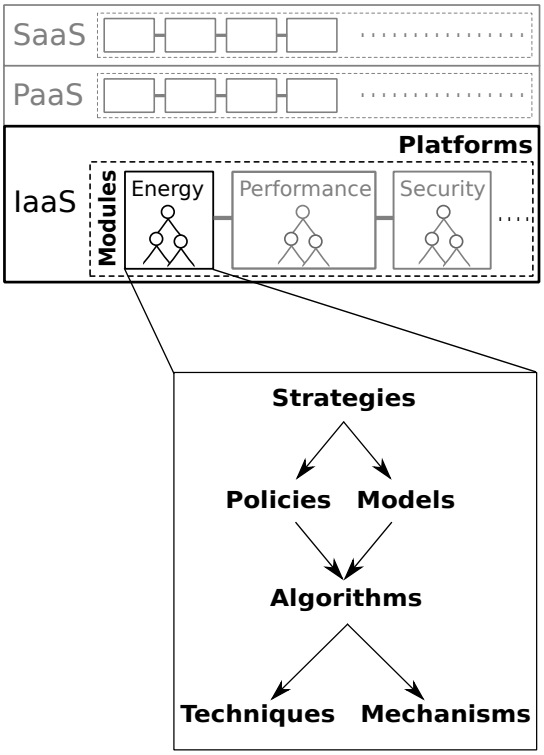


Figure 2.4 – It organizes the set of layers of components that meet the requirements by the cloud service provider.

Each of the terminology components is detailed below:

- *Mechanisms:* We define mechanisms as the set of special-purpose tools available in the hardware and operating system levels. Mechanisms are accessed through Applica-

tion Programming Interfaces (APIs) or programming libraries supported by the operating system to manage hardware features. In this sense, one can say that a mechanism is pre-existing in hardware and software, even if it is not used. Wu et al. [WCC14] discusses the DVFS mechanism, which is the support of modern CPUs to changes in its frequency by operating system governors, and consequently it may reduce processor voltage to save energy. Sleep states is another mechanism, presented by Min et al. [MWT+12] as an energy-efficient option, and it consists of replacing states with little energy saving, by a state of greater energy efficiency. Such mechanism supports a variety of states with many energy saving levels, each one with specific characteristics, and/or applied to individual components.

- *Techniques:* Techniques are a set of hardware-independent procedures abstracted from the operating system. In this event, the system may add capabilities on top of those already supported by hardware and operating system, and enable event handling to meet the goal of a strategy. Accordingly, it can be said that technique is purposely used as a method to achieve a target. Clark et al. [CFH+05] present the technique of virtual machines migration for virtualized environments, such as clouds. This technique consists in transferring services hosted in virtual machines from one physical server to another. The migration of virtual machines across different physical servers brings several advantages such as load balancing, online maintenance, fault tolerance, and power management. These benefits can increase flexibility and reduce the complexity of managing physical resources in such environments, besides reducing resource usage.

- *Algorithms:* Algorithms consist in translating an idea, a statement (called strategy in this classification), to a logical implementation, in other words, the realization in code form of policies, models, or both of them. The implementation of such algorithms manage policies and/or models for the purpose of adapt techniques and mechanisms of the infrastructure to meet a strategy. Han et al. [HGGG12] presents algorithms implemented on an IC-Cloud resource management system (RMS), which enables a cost-effective elasticity, based on utilization rates of the processor, memory, and network, to meet the quality of service requirements.

- *Models:* Models are mathematical formalisms, deterministic or stochastic, expressing relations, variables, parameters, entities, and relationships between variables and entities or operations, aiding decisions about resource management. Models are often used to estimate what the actual resource needs, depending on high-level metrics based on the applications or customers behavior. Pietri et al. [PJDS14] present a prediction model of cloud workloads behavior, based on the relationships among the amount of tasks, the minimum and maximum acceptable time for their execution, and some available slots. Khomonenko and Gindin [KG14] propose a multichannel non-

Markovian queue system that analyzes the performance of cloud applications, based on a number of available resources.

- *Policies:* Policy consists of a set of rules and norms that can determine actions through predetermined thresholds, which promotes a dynamic cloud environment through changes in the infrastructure to meet particular strategy. Additionally, policies may limit the decisions on the use of resources, helping to maintain a number of resources within the acceptable quality of service levels. Suleiman and Venugopal [SV13] analyzed several policies that determine the minimum and maximum use of some resources (processor usage) or high-level metric (response time), aiming to determine when operations of scale-out on cloud resources should be conducted. Therefore, policies consist of limits imposed by the service provider, and when these limits are exceeded, some action must occur.

- *Strategies:* Strategy is a solution used to achieve a module goal. In the context of cloud computing, a strategy is a way in which models and policies are applied to the available computing resources with the intention of improving some aspect of the cloud to better support services. Alvarruiz et al. [AdACH12] propose, as a strategy, shutting down idle hosts in a cluster to save energy. In a layer below, policies and models defined when such changes will occur, and what limits these changes must obey.

- *Modules:* Modules are related to non-functional requirements of the cloud environment [VS11]. The growing number of customers of cloud services makes non-functional requirements such as portability, performance, security and mobility, essential in most web systems. Furthermore, concerns about interoperability, geographic location, energy saving, and high availability are also part of the mapping and development of non-functional requirements. In this way, modules aimed at intelligent management of resources by the service provider, in addition to supporting the offered services to customers with the best quality of service. Each module has a particular purpose, and generally, there are several modules managed by the same platform, working concurrently and in some cases complementary to each other.

- *Platform:* Platform is a software layer that aggregates the management and the interconnection of capacities among cloud infrastructure modules. Each of these modules can be defined as a requirement to meet a high-level metric proposed by the cloud service provider. Therefore, platforms can handle one or more modules to enforce balance among the different cloud requirements. An example of infrastructure platform is OpenStack [RB14], a global collaborative project that develops an open source platform for management of public and private clouds. The project aims to provide solutions for all types of clouds, enabling simplified deployment, massive scalability, and easier management of resources. The technology consists of a series of interrelated projects that allow full implementation of a cloud infrastructure solution. Data and

virtual machines can be distributed across multiple data center hardware, being the OpenStack responsible for replication and integrity among clusters. In the same way, Eucalyptus [NWG+09] is an open source cloud platform that enables organizations to create a private cloud environment within its data center using an existing virtualized infrastructure. Eucalyptus enables private companies and organizations to build a compatible cloud with the Amazon Web Services, expanding the existing virtualized resources based on on-demand services.

2.4 Related Work

Modern operating systems and virtualized platforms bring, through the ACPI, opportunities for power management with the use of energy-aware strategies on idle hosts [CAT+01, HDC+05, ZQR+07]. The question of minimization of the operational costs by reduction of power consumption in cloud environments is widely discussed in current research, as shown by Gao et al. [GWGP13]. Isci et al. [IMK+13] show that there is an opportunity for energy-savings strategies in these environments using the concept of sleep states. Sleep states refer to the S-states of the ACPI specification and are adopted by strategies that "wake up" hosts from sleeping or hibernation states, bringing the system to an operational state.

Most solutions that promote energy savings in cloud environments have not only one of the terminology components proposed in this chapter. They are hybrid solutions that present various components at different levels, working together to support the needs of a particular module. In this section we show some of these work, and after we relate them to our proposal.

Some studies use sleep states to save power. Min et al. [MWT+12] present a strategy that selects the best sleep state based on typical workloads for smartphones. To switch from an idle state to another with lower power consumption, some policies (such as idle time and time in each sleep state) were used along with an algorithm that is applied to different states on the device. Results showed energy savings of up to 50%. Alvarruiz et al. [AdACH12] proposed a management strategy for clusters and clouds that save power by turning off idle hosts across the network. When the task releases a host, an algorithm uses a policy that checks the time of host in the idle state. This timer prevents the host to shut down when there is a possibility for a new job to run. When the timeout is reached, the host is turned off. Results showed energy-savings of 38% for cluster environments and 16% for cloud environments, respectively. Duy et al. [DSI10] presented a scheduling algorithm integrated with a model that uses neural networks to optimize the power consumption of servers in a cloud. The workload prediction is based on a historical usage. According to the model, the algorithm turns off unused hosts with the intention of minimizing the number of servers running, thus also reducing power consumption of the hosts. Evaluations showed

that this model can reduce power consumption by up to 46% compared to energy-agnostic environments. Maccio and Down [MD15] propose modeling sleep states for servers based on Markov Chains. Four states supported the model: off, setup, busy, and idle. The setup state is the transition time among the other three states. Through an incoming job guided by Poisson behavior, the model optimizes the states on multiple hosts to meet SLA restrictions.

Additionally, some studies use sleep states to save energy, along with solutions that aim not impact on performance. Niyato et al. [NCS09] proposed a power management strategy based on a Markov model to adjust the number of active servers for maximum performance. Although a considerable amount of power can be saved by shutdown and restart operations on hosts, the main goal was to perform the configuration of autonomous resources and enable online regulation according to the service behavior, power consumption, and SLA requirements. Results showed an increase of energy efficiency by up to 30%, minimally impacting performance. Zhu et al. [ZLLW12] proposed a strategy that split a cloud into four areas: busy, active idle, sleep, and shutdown. In the first area, the hosts are allocated to running applications. The second area retains a certain amount of hosts in the idle state, waiting to meet any possible demand. At the next level, hosts are kept in a state of suspension, ending with a level in which the hosts are turned off. This division provides an environment that classifies hosts into categories related to the environment usage. The results show that this organization can reduce power consumption of idle hosts in up to 84%, with an impact on the runtime of up to 8.85%.

Some studies use VM migration to save energy. Dong et al. [DZJW15] propose an algorithm that scales in a multidimensional manner the virtual machines on a homogeneous mobile cloud depending on two factors: the rate of CPU usage and the bandwidth among available hosts. Based on previous analysis, the minimum energy consumption and the number of physical machines in operation are derived. The results enable the development of an algorithm for virtual machines placement to save power.

Alboaneen et al. [APT14] present a VM placement strategy aimed at saving energy, but also with concerns about application performance. VM consolidation is based on the utilization rate of hosts that support applications, in addition to the available bandwidth between these hosts. Based on these values, the authors predict the workload behavior and decide the amount of VMs that each host must support every moment, guiding the VM placement. In the same way, Shen et al. [SLD+15] used a Markov model to allocate virtual machines on hosts to save energy, aiming to improve the trade-off between performance and energy savings. Compared with the state of the art suggested at work, the proposal achieves 23% energy savings.

Beloglazov and Buyya [BB10] presented an algorithm for virtual machine allocation in the cloud with the goal of saving power. Using a policy, such algorithm determines when and what virtual machine should be allocated on available resources with a minimum amount of migration to reduce the overhead and to avoid SLA violations. When virtual ma-

chines are migrated, idle hosts may enter into a sleep state, reducing thereby the overall power consumption. Results showed power savings of up to 83% compared to energy-agnostic scenarios, although they showed a minimal SLA violation. Lefèvre and Orgerie [LO10] showed a cloud strategy that saves power due to several factors, such as startup and shutdown hosts, control the usage rate of resources by policies, and uses the migration of virtual machines technique. An algorithm to predict through a model the behavior of the workload has been proposed. The experimental results showed differences in power consumption among the various scenarios (an algorithm that turns on/off hosts, an algorithm that migrates virtual machines, or a mixed algorithm). In these experiments, energy savings of up to 25% was achieved when compared to an energy-agnostic environment. Moreover, the results showed that depending on the type of resource, the best option might vary. Feller et al. [FRM+10] proposed a consolidation model for workloads coupled with a dynamic adaptation of sleep states and changing processor frequency. The paper presents a part of the proposed model, whose goal was to minimize the number of hosts that host applications. For this, a model for adjusting the workload on the hosts and setting policies to control the transition between idle and off hosts was presented. The paper proposes only a model and lacks of a validation versus a real-based environment.

Santana et al. [SLM10] proposed a model for predicting the behavior of applications on web clusters, aiming to apply DVFS and turn idle hosts off, trying to keep up the quality of service. The metric assessed was the rate of processor usage. Results showed an energy-saving of up to 59%, trying to keep up the quality of service in 95%. On several occasions, this quality of service could not be maintained precisely due to the action of turning off and restarting hosts. Ding et al. [DQLW15] presents a virtual machine allocation algorithm on cores with different frequencies. Within certain periods, this organization is performed again, making the environment to self-adjust always aiming energy savings. Through simulation, the authors claim that their strategy can save up to 20% energy. However, the paper assumes that the change in frequency is performed individually on cores where virtual machines are attached, but this operation is not common in most modern processors.

Krioukov et al. [KMA+10] proposed a strategy for heterogeneous clusters with a focus on saving power and the least possible impact on the response time of the tasks. Three different architectures were simulated: Nehalem, Atom, and BeagleBoard, and as a workload trace, seven days of HTTP traffic of Wikipedia were used. DVFS have been used over underutilized hosts, and sleep states on idle hosts, besides shut down a slice of available hosts in the cluster. The decision on the ideal amount of hosts to meet the tasks is based on combinatorial optimization model (knapsack problem), and the results showed an energy-saving of up to 27%, with less than 0.03% of lost requests. In clusters, the use of DVFS is not recommended because when the frequency of the processor is reduced, the number of instructions that can be performed are too. The trace used had a low usage rate, and probably because of this, the results were so satisfactory. Employing workloads

with higher rates of use or a queue of tasks with dynamic arrival rates, the response time is likely to be significantly affected, besides the environment suffer an increase in power consumption.

Table 2.2 – Summary of related work.

Work	VM consolidation	Sleep states	DVFS	Performance
Min et al. [MWT+12]	No	Yes	No	No
Alvarruiz et al. [AdACH12]	No	Yes	No	No
Duy et al. [DSI10]	No	Yes	No	No
Maccio and Down [MD15]	No	Yes	No	No
Niyato et al. [NCS09]	No	Yes	No	Yes
Zhu et al. [ZLLW12]	No	Yes	No	Yes
Dong et al. [DZJW15]	Yes	No	No	No
Alboaneen et al. [APT14]	Yes	No	No	Yes
Shen et al. [SLD+15]	Yes	No	No	Yes
Beloglazov and Buyya [BB10]	Yes	Yes	No	No
Lefèvre and Orgerie [LO10]	Yes	Yes	No	No
Feller et al. [FRM+10]	Yes	Yes	No	No
Santana et al. [SLM10]	No	Yes	Yes	Yes
Ding et al. [DQLW15]	Yes	No	Yes	No
Krioukov et al. [KMA+10]	Yes	Yes	Yes	Yes
E-eco	Yes	Yes	Yes	Yes

Table 2.2 summarizes the studies presented in this section to enable better visualization of the used energy saving approaches, as well as identification of any bias aiming application performance concerns. We can see that only the Krioukov et al. [KMA+10] manages VM consolidation, sleep states, and DVFS the same time. However the focus of the work is HPC clusters. Besides, some studies analyze application performance, although they prioritize energy saving over application performance.

Some discussed works provided a basis for the development of our work. Initially, based on Alvarruiz et al. [AdACH12], we evaluated the cost of transitions between different states of suspension, given that the proposal presented by the researchers only turning on/off the hosts, which affected the execution time of applications, especially in HPC clusters. The work of Zhu et al. [ZLLW12] proposed split the cloud environment in several sites with different states of suspension. However, our evaluations demonstrate that the cost of transition between states and energy savings between some of them were very close, and thus there is no direct benefit to using them. The work of Santana et al. [SLM10] introduced the DVFS as a proposal for energy savings in clouds, which raised one of the questions addressed in our work, which is the decision between reducing the processor's frequency and keeping the environment as it is or consolidating VMs and turning off idle hosts.

Although all these solutions address applications performance issues and energy-saving possibilities in computing environments, neither assesses their mutual impact or a

solution to improve such a trade-off. Thus, we believe that a solution to manage the two discussed energy-saving options bounding its utilization to the applications performance can reduce this trade-off. On the contrary of previous work, there was not a work developed with particular focus on the trade-off between saving power and application performance.

2.5 Concluding Remarks

The concept of cloud computing refers to the use of memory, and capabilities for storage and calculation from shared computers and servers. These resources are available and interconnected through the Internet, and all the stored information can be accessed remotely from anywhere, at any time, on demand, based on a pay-per-use model.

This new model has quickly become a standard model for offering services to large data centers, which boosted the growth of these environments to support the growing customer demand. Therefore, the greater the amount of customer requests on services, the higher must be the data center, the more energy is needed and consumed. Due to current environmental issues, there is a need to reduce the energy consumption of large corporations, and consequently, data centers are also the target of such reduction.

There are several solutions designed to reduce energy consumption in such large-scale environments. The aim of this chapter is answering research question 1, which is to determine the support offered by hardware and software so that techniques and mechanisms can save energy in cloud environments. Therefore, this chapter discusses cloud concepts and ACPI standard that supports power-saving states. As a contribution to this chapter, we proposed a bottom-up terminology to unify the terms used in this study, and also to concede organization and a better understanding of the solutions presented by the state of the art.

The knowledge acquired in this chapter allowed to know techniques and mechanisms that can be applied in computing environments aimed at saving energy. As the planning of this work, the following chapter examines the impact of such techniques and mechanisms on a host, serving as the basis for the next experiments.

3. INTRA-HOST PERFORMANCE AND ENERGY-SAVING TRADE-OFF ANALYSIS

Smarter power management becomes essential in all computational systems to achieve better results regarding green computing and also reducing costs on a given amount of computational work, especially in large data centers that host thousands of hosts [MNDR10]. In doing so, a large variety of models has been developed to draw a threshold line between energy efficiency and power consumption. These models have focused on components that spend more power, such as CPU, memory, and disk.

Specifically, when dealing with power consumption of the processor, there is an opportunity to save power by using the DVFS (Dynamic Voltage and Frequency Scaling) [LKLK09], which is a feature of the modern processors that allows frequency is changing. Linux kernel permits the use of several governors, which allows the adjustment of processors frequency to the applications behavior. In this way, several studies show the benefits of DVFS at different operating governors using several different types of scenarios. DVFS is an interesting feature to save power. However, it's hard to predict the consequences of DVFS or even verify its feasibility for power-saving, besides the difficulty deducing which parameters work better in each case. Mathematical models allow an approximation to real results, without any actual measurements involved.

Besides, while most previous studies are related to at least one ACPI state, none of them explore the cost to go from one state to another. For example, if a given host goes into the G2 state, that is, it is turned off to meet an energy-saving strategy, how long does it take to turn back on? And how much energy is spent during reboot? Because the transition time between states has a direct impact on the application performance, they need to be considered if one aims to build accurate models that aim at increasing energy efficiency with minimum application performance. This goal, this chapter presents evaluations of DVFS and sleep states on a host, taking into consideration the impact of each option on the trade-off between performance and energy savings.

The models for single hosts developed in this chapter form the basis to more sophisticated models that consider multiple interconnected hosts and clouds that are discussed in the next chapter of the dissertation.

3.1 Modeling DVFS governors

In this section, we present multiple linear regression models for power consumption estimation based on DVFS and CPU usage. This model enables profiling resources and power consumed by a machine, under the main governors of DVFS. The regression

model was fitted to a set of results based on the SPEC benchmark. To evaluate the proposed model, GROMACS [vdSvMC12], a molecular dynamics application, was used in high-performance environments. During its executions, power consumption, CPU load, and frequency were collected and compared with estimated values from model's regression.

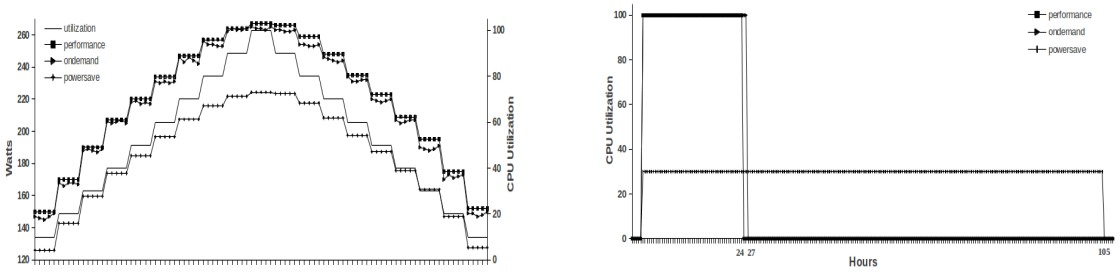
The main contributions of this chapter are (i) an evaluation of DVFS governors under a linear regression; (ii) a mathematical model for measuring the power consumption of the nodes; and (iii) a model validation using a well-known industry standard.

3.1.1 Modeling Evaluation

To create an initial model, data was collected from running servers. The power consumption is acquired by using a multimeter connected between the power source and the machine host. This device (EZ-735 digital multimeter) has a USB connection that allows periodic external reading and provides the values of power consumption in watts.

Once this device was connected, the experiments were performed on a node with two Intel Xeon E5520 (16 cores total), 2.27GHz, 16 Gb RAM with Ubuntu Linux 12.04 (Server edition). In the server edition, DVFS is set to *Performance*, and this leads us to assume that systems developed for large-scale or HPC environments are configured to keep the maximum possible performance, without concern for power saving.

The benchmark used in model fitting was SPECint (Standard Performance Evaluation Corporation) [BL12], a well-known industry standard benchmarking tool. SPEC benchmarks are widely used in intensive computation to evaluate different architectures for high performance, enabling comparison between them. Specifically, SPEC is used to assess the processor performance, through a CPU-intensive application. The model fitting details are explained in the next section.



(a) a (b) b

Figure 3.1 – Primary Evaluations. (a) Workload Behavior vs. Power Consumption; (b) CPU usage vs. Execution Time

Figure 5.9 shows the behavior of the workloads, which are controlled slices of processor usage between 10% and 100%, and their relative power consumption in watt-hour. Evaluations using different rates of processor usage are justifiable because it represents the

behavior of most production applications. At 10%, the tests show typical behavior of IO-bound applications which spend the most of the time executing input/output. At 100%, the tests show the typical behavior of CPU-bound applications which spend more time using the processor. Figure 5.10 shows the relationship between the CPU usage for the three DVFS governors and the execution time in each of them.

With the analysis of such behaviors, we observed that there is a linear relationship between power consumption and CPU usage, and a relationship between the DVFS governor models and the execution time of jobs. This approach has been used in other studies as in [BB12]. By doing this, Table 3.1 present the results of DVFS *ondemand* governor measurements as well as the frequency and execution time related to each rate of CPU usage. It is possible to infer that the use of *ondemand* governor causes a fluctuation between the frequencies of the processor, which provides a power consumption adjustment, impacting on the execution time. Table 3.2 presents the measurements of power consumption and execution time of the rates of CPU usage when the *performance* governor is used. The results show that the processor is always running at its greatest frequency, which reduces the execution time, but increases power consumption. The *powersave* governor is an alternative to reduce the power consumption. Table 3.3 shows the results of this governor, which always uses the lowest frequency possible. Although it has lower power consumption per second, it has a significant impact on the execution time of the application.

Table 3.1 – Ondemand governor measurement results.

CPU %	Frequency	Time (sec.)	Watts
10	1.6	205200	0.041
20	1.6	190800	0.047
30	1.73	176400	0.052
40	1.73	165600	0.057
50	1.86	151200	0.061
60	1.86	136800	0.065
70	2	126000	0.068
80	2.13	111600	0.071
90	2.13	97200	0.073
100	2.26	86400	0.074

Such measurements confirm the expected trade-off between power savings and execution time. Although the execution time when running in ondemand governor is a two goals function, related to processor usage and processor frequency, performance and powersave governors are not. To develop a more general model, we chose to work only with one dimension. The model proposed in this section could be used to infer these values, based on the usage rate of the processor.

Due to there are independent variables (Tables 3.1, 3.2, and 3.3) that maintain a relationship between itself, a multiple linear regression models appears to be a suitable model to estimate the values of the dependent variable.

Table 3.2 – Performance governor measurement results.

CPU %	Frequency	Time (sec.)	Watts
10	2.26	203000	0.052
20	2.26	188500	0.054
30	2.26	174100	0.057
40	2.26	163100	0.060
50	2.26	149300	0.063
60	2.26	134700	0.067
70	2.26	124400	0.070
80	2.26	110800	0.072
90	2.26	96800	0.074
100	2.26	86400	0.074

Table 3.3 – Powersave governor measurement results.

CPU %	Frequency	Time (sec.)	Watts
10	1.6	205200	0.041
20	1.6	206300	0.042
30	1.6	201500	0.043
40	1.6	194700	0.045
50	1.6	189100	0.046
60	1.6	185200	0.048
70	1.6	181600	0.051
80	1.6	176100	0.054
90	1.6	169900	0.057
100	1.6	165000	0.061

This regression is the base of the following model ($F - statistics = 19, P - value = 0.05$):

$$P_{(watts)} = B_0 + B_1 P_{(frequency)} + B_2 P_{(time)} + B_3 P_{(use)} \tag{3.1}$$

where $P_{(watts)}$ represents the total power consumed in the entire host. The parameter B_0 represents an initial state without power consumption. The parameters B_1, B_2 and B_3 are the weights assigned to each variable, in each DVFS governor. $P_{(frequency)}$, $P_{(time)}$ and $P_{(use)}$ are monitored values that were presented in such Tables. For each set of data in these three tables is generated different weights for each variable in the model. An important validation aspect of a set of multiple linear regressions is the residue analysis, which shows the model significance and evaluates the contributions of regression variables. The accuracy of estimation of this model is greater than 94% when compared against the external measurement method using the multimeter.

To evaluate the model, nine real traces for different CPU usage rates from molecular biology software were used. Molecular dynamics simulations are one of the main methods used in the theoretical study of biological molecules. This computational method calculates

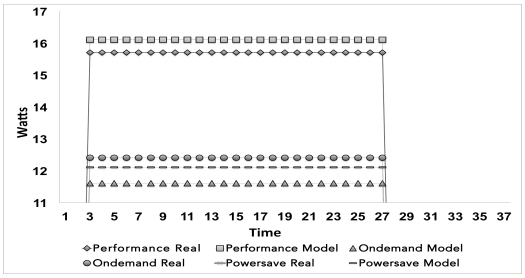
the behavior of a molecular system over time. Molecular Dynamics simulations have provided detailed information on the fluctuations and conformational changes of proteins and nucleic acids. GROMACS (GROningen MAchine for Chemical Simulations) [vdSvMC12] is a molecular dynamics simulation package and has been used in several studies.

The algorithm simulates each time step by calculating the atom force fields and solving motion equations. These motion equations are based on the acceleration obtained from previous time step forces and functions of prediction and correction of parameters (e.g. pressure, temperature, acceleration, etc.). We used 24 hours traces (*performance* governor), with different rates of CPU usage: 5%, 16%, 25%, 37%, 49%, 56%, 66%, 75%, 83%. Such different utilization rates are different phases of the same trace execution. The differences in CPU usage rates of these traces are intended to test our model on different scenarios, allowing the verification of its behavior in each one. To draw the trade-offs between DVFS operating governors, the same evaluations were performed with the three main governors: *performance*, *ondemand*, *powersave*. The other two governors (conservative and userspace) are modified versions based on one of these three main governors, which are not usually used in production.

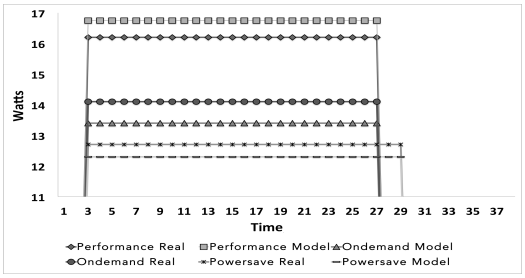
3.1.2 Experiments and Discussion

This section presents the evaluation of the mathematical model against nine real traces of executions of GROMACS using three DVFS governors. The tests confirm estimates generated very similar results compared to the real power consumption of each trace. An important point to be considered is the power consumption of the *powersave* governor. In a small CPU usage scenario (up to 25% average usage, Figures 3.2 (a), (b) and (c)), this governor does not have a significant impact on the execution time when compared to the other two governors (*ondemand* and *performance*). It means that the *powersave* governor is best suited to the behavior of this usage rate, consuming less power, with minimal impact on the execution time.

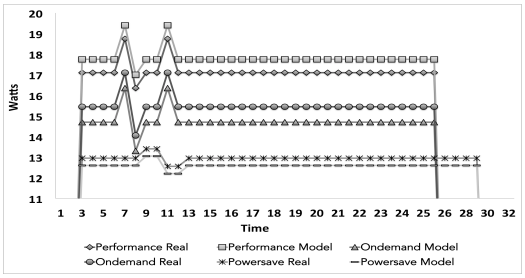
By analyzing the charts with usage rates between 37% and 49% (Figures 3.2 (d) and (e)), we can see that there is a better balance between power consumption and execution time when the *powersave* governor is applied. Still, there is an increase of 50% in execution time, with a reduction in power consumption of only 30%. For above 50% of CPU usage (Figure 3.2 (f), (g), (h), and (i)) the *powersave* governor, although saving more power each time slice, significantly increases the execution time compared to the other two governors, consuming more power as a consequence. *Ondemand* and *powersave* were governors that had greater power savings compared to the execution time. Although the *powersave* governor is idealized to consume less power when using high rates of resource usage, the execution time increases due to low frequency, which discourages its use in



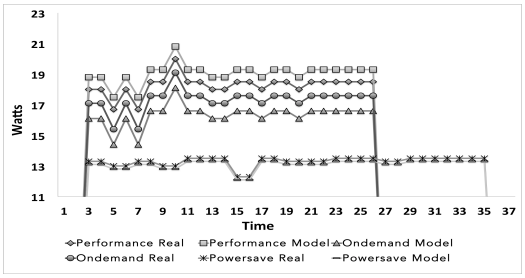
(a) 5%



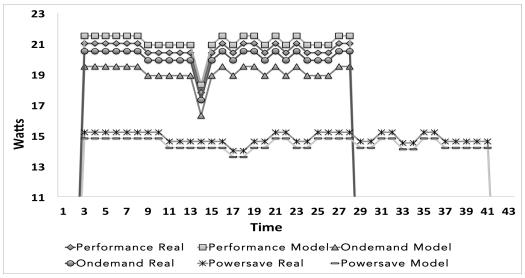
(b) 16%



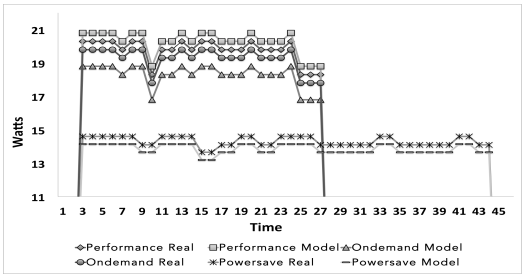
(c) 25%



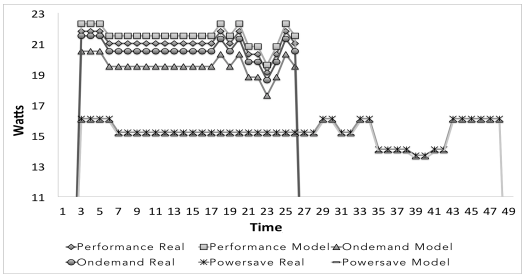
(d) 37%



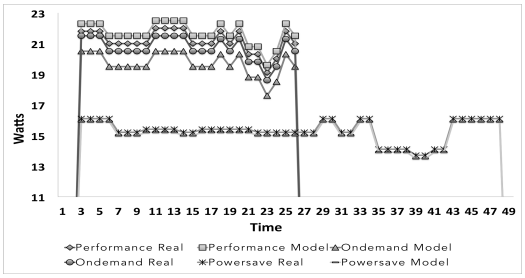
(e) 49%



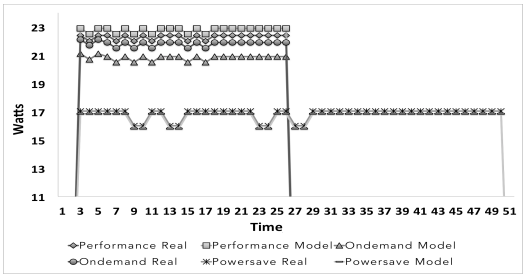
(f) 56%



(g) 66%



(h) 75%



(i) 83%

Figure 3.2 – Traces Evaluation

this case. Unlike the behavior is shown in tests with low processor usage, this set of tests is focused on the behavior of processes that perform processor-intensive, or CPU-bound jobs. As a higher rate of processor usage, it is expected that there are significant differ-

ences among DVFS governors. The greatest impact between *performance* and *powersave* governors show the trade-off between power savings and performance.

3.2 Modeling the ACPI Sleep States

Figure 3.3 depicts power-agnostic states in a host and their transitions from a holistic view of a data center, where the requested hosts are *busy* and unused hosts are *idle*. Under these constraints, it is possible to infer several policies to decide on conditions to enter or leave ACPI states to save energy. However, the trade-off concerning energy consumption and performance to change power states are not considered in current works, even though they are fundamental to answer the above optimization questions.

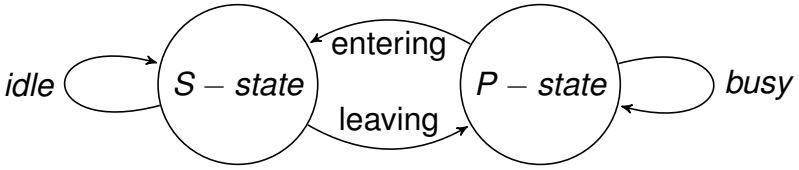


Figure 3.3 – Power-agnostic states of a host.

From the ACPI's point of view, when a host becomes *busy*, its CPU might be put into one of the load-driven power states (P-states). On the other hand, the host might enter into an S-state when it becomes idle, reducing the energy consumed by hardware components in both states. This is a very typical scenario in laptops and mobile devices to conserve energy. In a more generic sense, the set of ACPI G-/S-states can be correlated as shown in Figure 3.4.

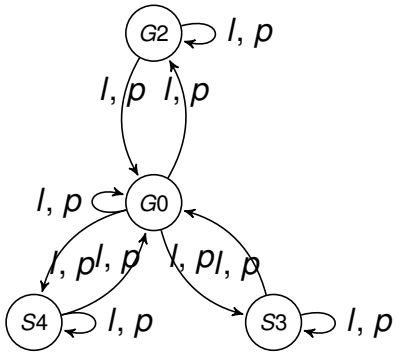


Figure 3.4 – Correlation of ACPI S-/G-states. Edges represent reachability between states and weights the latency and power rate to switch between them.

It is worth noting that all states at some point converge at the G0 state when the host is busy in processing a workload. It occurs because any transition must pass through G0, as per the ACPI specification. The weights *l* and *p* denote, respectively, the latency (seconds) and power rate (watts) required for a state to be reached. Moreover, the power rate while the host remains in each state is represented as a transition to the state itself.

Although transitions between states can be measured at discrete time intervals and are represented by different reachable states, they do not present a probabilistic behavior. State transitions are deterministic based on well-defined and controlled events within a limited set of states. This deterministic behavior of ACPI's state transitions makes them unsuitable to be modeled via stochastic processes, such as Markov Chains. Therefore, according to graph theory we express this behavior as $G = (V, A)$, where $V = \{\nu_i, ..., \nu_n\}$ is the vertix-set, and $A = \{\epsilon_i, ..., \epsilon_m\} \subseteq \{(x, y), \{x, x\} \mid x, y \in V\}$ is the edge-set. From the ACPI standpoint, we define: $V = \{G0, G2, S3, S4\}$ and $A = \{(G0, G2), (G2, G0), (G0, S3), (S3, G0), (G0, S4), (S4, G0), \{G0, G0\}, \{G2, G2\}, \{S3, S3\}, \{S4, S4\}\}$. Power ($p$) and latency ($l$) are the weights of each $\epsilon \in A$ such that $F(\epsilon) = ((x, y), \{x, x\})$.

To discuss our claims, we conducted a set of preliminary tests in a physical host to identify relationships between state changes under different load conditions. We assessed the power rate while the host was changing from/to G0, G2, S3, and S4 states. In the case of (G0,G0), in which the host is busy, and the CPUs might enter in a load-driven P-state, we scaled them up to the maximum frequency for peeking energy consumption. The testbed consists of a host equipped with two 2.27GHz Intel Xeon E5520 processors (with eight cores each), 8M of L3 cache per core, 16GB of RAM and one NetXtreme II BCM5709 Gigabit Ethernet adapter. The instantaneous power rates were measured via a digital power meter connected directly to the host's power supply and the latency was obtained by inspecting power fluctuation during transitions. The measurements for each state and its transitions are shown in Table 3.4.

Table 3.4 – Assessments of the power states and their transitions in a host. N/A means the host does not process any load in sleep mode. The symbol ∞ means the host may stay indefinitely in a state. Latency and Power are represented in seconds and watts, respectively.

ϵ	$l(\epsilon)$	$p(\epsilon)$	σ_l	σ_p
(G0,G2)	59	108	1.1	0.7
(G2,G0)	81	69	2.1	0.4
(G0,S3)	25	51	1.1	0.5
(S3,G0)	5	91	0.6	0.5
(G0,S4)	101	86	4.9	0.2
(S4,G0)	79	79	1.3	0.9
{G0,G0}	∞	190	N/A	N/A
{G2,G2}	∞	6	N/A	N/A
{S3,S3}	∞	9	N/A	N/A
{S4,S4}	∞	11	N/A	N/A

Latencies and power rates may vary during state transitions. Hosts with a large amount of data loaded in memory will probably take more time to be powered off than idle hosts. To reflect this, several measurements of latency and power were performed while the host was under different load conditions regarding memory and CPU usage, and the measures were based on the average with their respective standard deviations σ. The total

amount of memory allocated varied between 10% and 100% and the CPU load was increased in a core basis, starting from 1 to 16 using the Linpack benchmark [DL11]. The highest standard deviation was observed in ($G0$, $S4$). Data was moved from the memory to disk before entering the S4 state. This caused the latency to vary unpredictably because of the amount of data in memory. We believe this is not an issue because energy-saving strategies in most cases do not change host's power states until it is unallocated so that memory content is cleaned up and space is returned to the OS before any policy is triggered.

Based on our observations, the energy consumed by a host during a given state transition ϵ is calculated by the integral of the power rate using the first instant time t_0 and the amount of time the host is changing to a state $t_l(\epsilon)$ as limits:

$$E_\epsilon = \int_{t_0}^{t_l(\epsilon)} p(\epsilon) Dt \tag{3.2}$$

Given that a host may have its power state changed many times governed by an energy-saving strategy, then we need a discrete equation to sum the energy consumed by a set of executed transitions. Thus, let $S : S \subseteq E$ be a subset of transitions executed for a period. The total energy consumed by transitions in S is given by:

$$E_T = \sum E_\epsilon, \forall_\epsilon \in S \subseteq \{(x, y), \{x, x\} \mid x, y \in V\} \tag{3.3}$$

Additionally, we also considered the consumption while the host is in the G0 state; that is, executing some task. The CPU's frequencies may vary dynamically to conserve power in the G0 state by entering into a P-state. Thus, we added to our definition the well-known linear power model proposed by Chen et al. [CGV+11] which considers CPU usage (α) as input to predict power in P-states:

$$E_{\{G0,G0\}} = \sum_{i=1}^{U} \cdot \int_{t_0}^{t_l} \left[(1 - \alpha) P_{FreqIdle_i} + \alpha P_{FreqFull_i} \right] Dt \tag{3.4}$$

Where the CPU power rate while it is idle and full utilization are denoted by $P_{FreqIdle}$ and $P_{FreqFull}$, respectively. The integral limits represent the amount of time the host remained in G0, and U is the total amount of processing units. Finally, the total energy consumed by transitions in S including the $\{G0, G0\}$ transitions, is denoted by:

$$E = E_T + E_{\{G0,G0\}} \tag{3.5}$$

It should be noticed that a transition occurs whenever a request arrives, and a given host must return to G0 to serve it. Thus, let $W : W \subseteq S$ be a subset of transitions performed over this condition, w the user's workload and t its execution time. The equation that represents the total workload execution time when the requested host is not in G0 and

the user must wait for a transition ϵ before having its workload placed on that host is given by:

$$ET = t(w) + l(\epsilon), \epsilon \in W \tag{3.6}$$

Finally, the latency-related performance degradation incurred by transitions in W is given by:

$$L_T = \sum l(\epsilon), \forall_\epsilon \in W \tag{3.7}$$

It is evident that there is a trade-off between the total energy consumption and performance. The impact of this relationship on real-world scenarios now becomes much clearer. An in-depth study reveals that current energy-saving strategies do not consider this trade-off and that there are environments where these transitions would have an enormous influence on energy consumption and user's SLA, such as those that the turnaround time is critical and should never be exceeded.

3.2.1 Modeling Evaluation

All experiments were conducted on the machines presented in Table 3.5. They are equipped with heterogeneous processor architectures and different resource capacities.

Table 3.5 – Configuration of the machines in our testbed.

Host	Processor	Cores	Clock	Cache	RAM
1	Intel dual core E5200	2	2.5 Ghz	2 Mb	4 Gb
2	Intel core 2 duo E8400	2	3,1 Ghz	6 Mb	2 Gb
3	Intel Xeon E5520	16	2.7 Ghz	8 Mb	16 Gb
4	Intel core i7 3770	8	3,4 Ghz	8 Mb	16 Gb
5	Intel core i5 2400	4	3,1 Ghz	6 Mb	8 Gb

Among all of the hosts shown in Table 3.5, five distinct architectures vary in power rates, as can be seen in Figure 3.5. The trade-off formerly noted during the modeling steps now becomes more noticeable. The deeper the power state, the higher the latency to return from the state. In contrast, the deeper the power state, the lower the energy consumed in the state.

The measurements collected from the hosts in each state are shown in Table 3.6, and the measurements from the transitions can be seen in Table 3.7.

In addition to the energy consumption, Table 3.6 also shows the power rate in G0 when the hosts' CPU becomes idle or goes up to a full load. The consumption while idle is

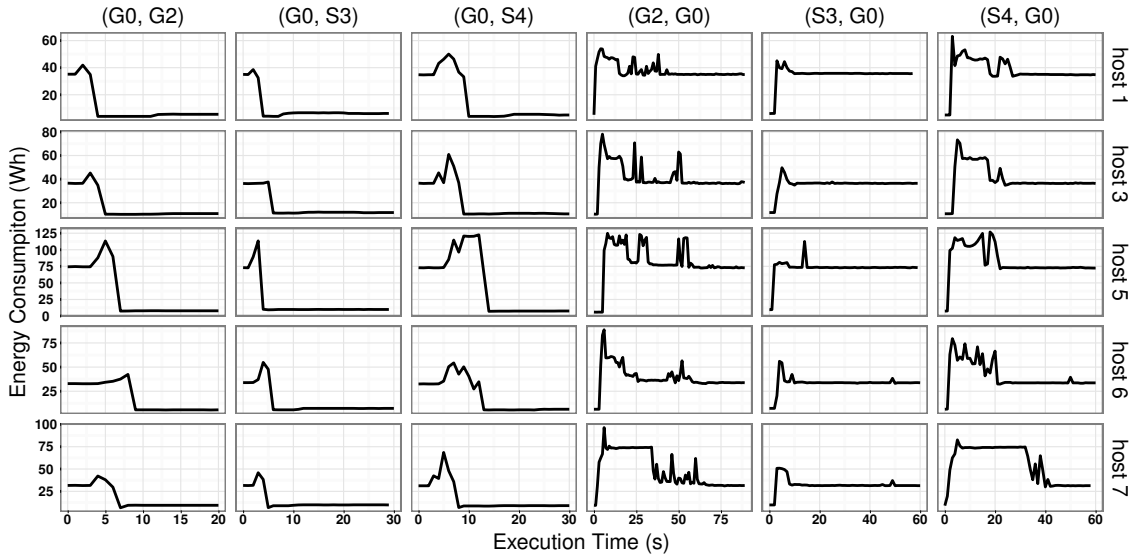


Figure 3.5 – Energy consumption during ACPI state transitions on different architectures.

Table 3.6 – Measurements of the energy consumption in each state for the five distinct hosts

	Power (W)				
	(G0,G0)		(S4,S4)	(S3,S3)	(G2,G2)
host	Idle	Full	*p*	*p*	*p*
1	32	70	12	15	5
2	36	82	8	11	5
3	110	190	24	12	5,5
4	32	95	9	13	6
5	31	92	10	13	6

Table 3.7 – Measurements of the energy consumption and latencies during state transitions for the five distinct hosts

	Latency (s) x Power (W)											
	(G2,G0)		(G0,G2)		(S3,G0)		(G0,S3)		(G0,S4)		(S4,G0)	
host	*l*	*p*	*l*	*p*	*l*	*p*	*l*	*p*	*l*	*p*	*l*	*p*
1	44	40	4	30	8	36	3	27	7	36	28	42
2	48	46	3	31	9	35	2	28	9	27	21	50
3	81	69	59	108	5	91	25	51	101	86	79	79
4	56	43	5	31	8	35	4	35	9	37	20	55
5	65	55	4	29	7	40	3	30	6	38	42	63

essential to reproduce strategies in which the host is unallocated and does not enter into a deeper power state immediately. On the other hand, the consumption while in the full load enables simulation of high-load host allocations. Decisions about energy-saving policies that lead to scenarios like these are influenced essentially by the trade-off we have pre-

sented. Furthermore, the latency values were suppressed just because a host can remain indefinitely in a state.

3.2.2 Experiments and Discussion

The trace-driven by a strategy that puts idle hosts into the three states (S3, S4 and G2) is presented in Figure 3.6.

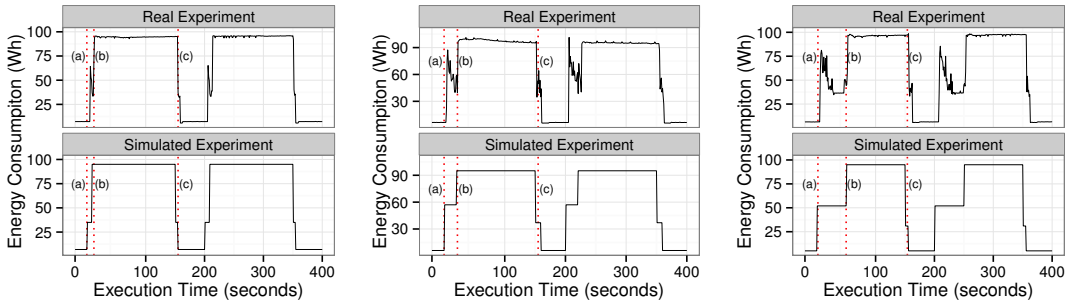


Figure 3.6 – Real and simulated energy consumption of all the three ACPI states: S3 (standby), S4 (hibernate), G2 (soft off).

In the beginning, the host was idle and remained in that G/S-state until a request for provisioning a workload was received. Slice (a) means there was a request, and the host must be ready to receive a workload. In this case, a transition to the G0 state has occurred, and the host started to leave its current state. While in G0, the host started the workload, and its processors are no longer idle (CPUs are at the maximum power peak since DVFS is in performance governor mode). This step refers to the slice (b) in the figures. Finally, slice (c) means the workload was finished, and the host became idle again, thereby, according to the energy-saving strategy policy, the host had its state switched back to either G- or S-state. Thus, it started transitioning into a state where the host consumed less power and remained there until the next request arrived.

The results showed that there are different impacts on an application execution time for each ACPI state. The deeper in energy savings the state is, the higher the latency in their transitions.

3.3 Concluding Remarks

Achieving high performance and low power consumption in modern microprocessors is a constant challenge in processor development. Many mechanisms for processor architecture require systemic understanding and an overview of the product, as these decisions' trade-off between performance and energy require much interaction with the software

and application. Dynamic Voltage and Frequency Scaling (DVFS) is a promising mechanism towards achieving energy-efficient computing in data centers. However, there are trade-offs among the different DVFS governors regarding performance and power consumption. Because frequency adjustments may have a negative impact on system performance, users often have to experiment with these governors to find the optimal configuration for their application and energy reduction goals.

Besides, the transitions between sleep states also lead to overhead.The latency of the transition from one state to another may take long enough to impact the application performance negatively. The deeper in energy savings is the state, the greater the latency of transition between one state and another. Based on this, the choice of the ideal states for each environment should be a wise decision, if the intention is to improve the trade-off between energy savings and performance.

The aim of this chapter was answering research question 2, discussing the impact of energy-saving options provided by ACPI on a host. These analyses provided support to the decisions of the following chapter, where we propose strategies for improvement in the trade-off between performance and power saving to a set of hosts, through the choice of states that are best suited to this objective.

4. CLUSTER PERFORMANCE AND ENERGY-SAVING TRADE-OFF ANALYSIS

Clusters are inter-host architectures composed of multiple computers connected by a network, working to solve massive problems in units of time which are impossible in conventional machines. In such clusters, the speedup is the main performance metric adopted to measure how much a parallel application is faster than a corresponding sequential application. However, with the grown of exascale machines[TLCS12], other metrics regarding energy saving have emerged in the past few years [KJF11], since the energy costs for cooling servers have been a major concern in large-scale data centers in developed countries [ESSA+12].

In such a way, there have been several studies with the purpose of identifying the trade-off between performance and power consumption in clusters [MNDR10]. Such studies are focused on the waste of unused resources which is commonly observed in educational institutions clusters [SS05] [LGW05] [Med05]. A closer inspection on such clusters reveals a usage rate that goes from 13.8% to 36.3%. As we can see, there are idle hosts in certain periods of time that consume power needlessly, a situation that could be avoided if they were better managed by using energy-efficient strategies. For instance, by changing the frequency of processors, as well as improving task scheduling [SS12], even for hosts that are in sleep states [HS08].

Although these strategies allow saving energy when dealing with such environments, other metrics should be taken into consideration. Strategies using sleep states incur additional latency in resource managers' job queues since requested resources may take longer to become available for allocation. The total time taken between the submission of a job to the queue till it is scheduled and sent back to the user is called job turnaround time, and it is a key metric used by resource managers such as PBS/TORQUE[1].

Most studies in this direction have been performed on low-throughput clusters because it uses a significant amount of idle hosts allowing to implement multiple power-saving strategies. On the other hand, it is likely that in these circumstances the job turnaround time suffers a direct impact. This impact reflects on resource managers of high-throughput clusters since the job queue might rapidly increase due to the time needed to turn the hosts on. Furthermore, high-throughput clusters tend to increase the energy consumption when their hosts need to be frequently turned off/on.

In this chapter, our hypothesis is that we can choose the best energy state to be applied to the cluster depending on its usage rate. Also, in most of the energy-efficient works [LZ10], it is possible to see that different architectures also have different characteristics regarding hardware information management, making it difficult to carry out handling systems

[1] http://www.adaptivecomputing.com/products/open-source/torque/

for sleep states. Furthermore, such works do not consider job queues. This is an important point that should be taken into account to avoid wasting time in the changes of states, in both time and power.

This chapter concerns on the advantages of using energy-efficient strategies for clusters that verify jobs in a queue that will be submitted to the hosts before turning them off, avoiding reverting states. The findings included in this chapter are: (1) an analysis of the impact of sleep states in job turnaround time and power saving for different workloads; and (2) a new strategy that checks the job queue before putting idle hosts into a sleep state and chooses the best strategy to use, depending on the cluster usage.

4.1 ACPI States Evaluation on Cluster Architecture

Preliminary experiments were conducted to investigate the behavior of the job turnaround when sleep states and DVFS are applied on idle hosts, considering different cluster usage rates. To do so, evaluations were performed with usage rates varying from 10% up to 90% using all states presented in Chapter 2. By these rates, it was possible to find the trade-off between power consumption and resource manager performance. An analysis of this behavior provides information for the development of the new energy-aware strategy.

To verify the trade-off between time and power consumption from all power-saving states transitions, we developed an energy module in the SimGrid simulator [CLQ08]. We simulated a cluster composed of 128 hosts. The based server host for the simulation consists of two Intel Xeon 2.2Ghz (each processor has two cores), 2GB of RAM and one Gigabit Ethernet adapter. Cluster resources were managed by Conservative Backfilling (CBF) [Wei98]. The difference here lies in the addition of the new power consumption module.

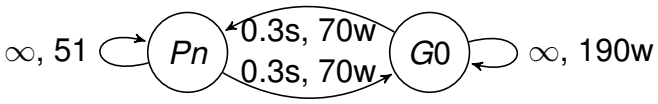


Figure 4.1 – The lowest DVFS P-State Transition

Complementing the measurements shown in Figure 3.4, Figure 4.1 presents transitions measurements between the G0 state and the processor in the lowest frequency. In our evaluations, we decided to ignore DVFS, due to the proximity of the values of time and power shown between DVFS and idle state based on Figure 3.4 and Figure 4.1 comparison. Moreover, this decision is also supported by the fact that the simulated environment comprises an HPC cluster, which should keep the hosts with the highest possible performance. Thus, only the states S3 (standby), S4 (hibernate) and G2 (power off) are considered.

Measurements of each transition were retrieved from a single host and were used to calibrate the simulator. These values were measured via a digital multimeter connected directly to the host's power supply, for monitoring the current during each transition. Based on these values, time and power consumption can be added or related to each newly submitted job to the simulator. Hence, this new module can show, at the end of each simulation, information on job turnaround time and power consumption for each set of HPC traces.

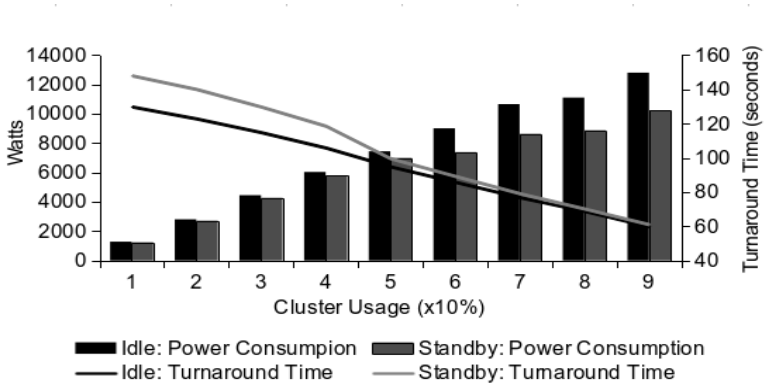
4.1.1 Preliminary Experiments

This section presents experiments on the behavior of the states without concerning the scheduler's job queue. After this analysis, experiments were performed through our strategy that checks the scheduler's job queue looking for jobs that have some hosts requested sufficiently compatible with the number of hosts that will be placed in one of the sleep states, keeping them turned on to receive the job. The experiments were performed with the power module developed to the SimGrid simulator, which allows simulation of power consumption of parallel and distributed applications running HPC environments, using several job schedulers and traces.

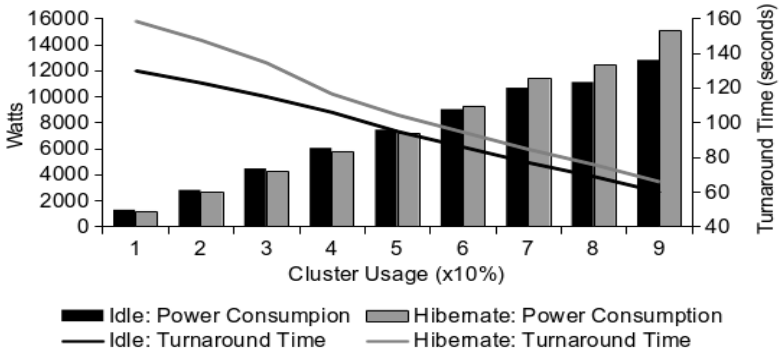
To simulate a stream of job submissions by users in an HPC cluster, we have applied two different approaches. The first was to simulate a synthetic workload based on a widely used model by Lublin et al. [LF03], which is one of the most comprehensive and validated batch workload patterns in literature. It uses two gamma distributions to model the job inter-arrival time (depending on the time of the day), a two-stage uniform distribution to model the job sizes and a two-stage hyper-gamma distribution to model the job execution time. This synthetic workload generator allowed us to develop sets of traces with different cluster usage rates. We chose to use ranges among 10% and 90%, which would enable us to visualize the points of variation in power consumption. In the second approach, we used real-world workload traces obtained from the Parallel Workloads Archive[2]. This archive has record information about the workloads on parallel machines, such as HPC clusters. We chose traces from the San Diego Supercomputer Center SP2 (SDSC SP2) [UTCL12], which is a well-known and widely studied workload. SDSC SP2 workload has 128 hosts and 73.496 MPI jobs, spanning two years from July 1998 to December 2000. In the evaluation, we use four slices of 24 hours each (Trace 1, Trace 2, Trace 3, Trace 4 of SDSC workloads), with usage rates of 55% of the cluster.

Figure 4.2 presents the preliminary tests we performed to view the execution time and power consumption of selected states, concerning the cluster usage. The first three figures used a statistical workload model based on Lublin et al. [LF03], while the last two figures used four real traces of SDSC. The standby state is shown in Figure 4.2(a) presents

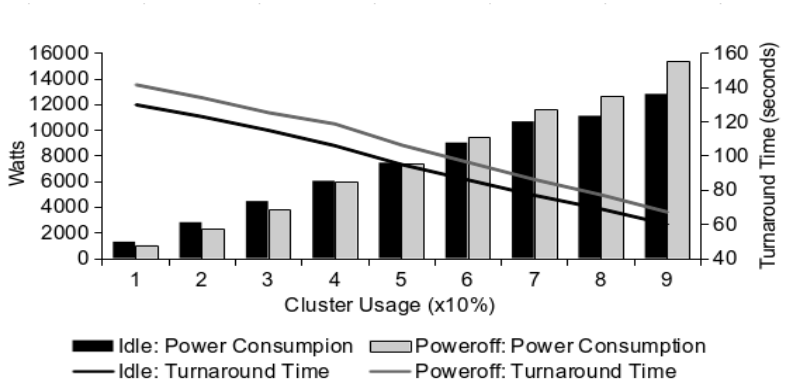
[2]http://www.cs.huji.ac.il/labs/parallel/workload/logs.html



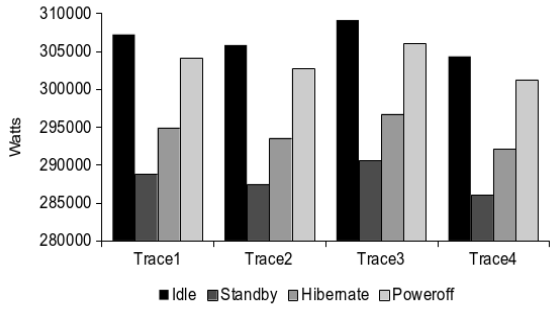
(a)



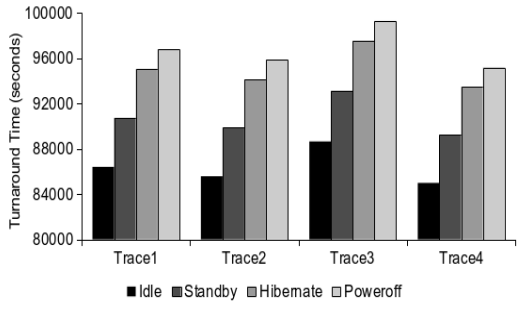
(b)



(c)



(d)



(e)

Figure 4.2 – Preliminary Evaluations

the best relationship between power consumption and execution time, when the use of the cluster is above 50%. This explains why, if we have a high usage rate, the entry of new jobs is more intensive, forcing the hosts at the cluster to change the state faster. As the ratio of power consumption and time to going to and leaves of each state is lower in the standby state, the results are satisfactory above this limit. The poweroff state can be seen in Figure 4.2(c), and it presents a better relationship between power consumption and execution time when the cluster is at low use, up to 30%. As clusters with low use keep a significant amount of idle hosts, and the entry of new jobs is not so intensive, the act of turning off the hosts can save enough power, while the time to restart these hosts does not impact the total time execution of jobs significantly. The hibernate state is positioned between these two thresholds shown, showing a better relationship between power consumption and execution time of 30% to 50%. As a cluster of average use, this state can balance the best way, issues such as higher entry jobs in the cluster take into account idle hosts at the same time. Figures 4.2(d) and 4.2(e) show the four real traces rate of 55% of use. The real traces confirmed the tests with synthetic traces, exhibiting the same behavior for this track usage of the cluster. As the results show well-defined thresholds, we can use these values to create a Best-fit Energy-Aware Algorithm that allows consuming less power, with less impact on the execution time of jobs. These thresholds are summarized in Table 4.1.

Table 4.1 – Usage Percentage of Energy States

States	**Usage Rate**
poweroff	30%
hibernate	50%
standby	over 50%

4.1.2 Best-fit Energy-Aware Algorithm

Several power-saving scheduling strategies for clusters are proposed in the literature. The focus of those strategies is usually on metrics such as performance and resource efficiency. The higher the resource usage, the higher the power consumption. Therefore, there is a high demand for technologies that allows supporting applications' performance, or losing performance at a rate that will not influence the response to the user, and save energy. This section is divided into two parts. The first algorithm shows the way our strategy allocates resources and uses the states of sleep mode. The second algorithm presents the second part of the strategy proposed in this Chapter, which uses all previous states together.

Algorithm 4.1 was developed taking into account the behavior of Conservative Backfilling (CBF) [UTCL12] scheduler. The CBF algorithm enables backfilling, and it is a well-known representative algorithm running on deploying RMS schedulers today. The main

idea of CBF is that an arriving job is always inserted in the first free slot available in the scheduler's queue, which offers an upper-bound to the job start time. Every time a new free slot appears, the scheduler sweeps the entire queue looking for jobs that can be brought forward without delaying the start of any other job in the queue. The CBF makes better use of available resources since it can run jobs out of arrival order. Such scheduler prioritizes jobs in the queue based on a number of factors such as priority. The key determinant for this scheduler to be reliable for HPC cluster environments is that each job has a start time and a time limit for execution. This means that the completion time of all jobs in the queue is known, and cannot be increased. Furthermore, CBF allows jobs to be run early in free slots since it does not delay other jobs.

At this time, the algorithm operates by checking if there are enough hosts to support the job execution. If there are enough hosts, the job is put into execution, and the unused hosts are put into a sleep state. If the algorithm detects that there are not enough hosts to meet job execution, enough hosts to meet the job request are awake.

Algorithm 4.1 CBF Improvement

```
numhosts ← hosts_requested_by_a_new_job
freeSlotList ← getFreeSlots()
for ∀freeSlot ∈ freeSlotList do
    slotDuration ← getSlotDuration(freeSlot)
    execTime ← estimateJobExecutionTime()
    if execTime <= slotDuration and numhosts <= freeSlotList then
        scheduling job
        sleeping unused hosts
    end if
    if execTime >= slotDuration or numhosts >= freeSlotList then
        awakening enough turned off hosts
        scheduling job
    end if
end for
```

Algorithm 4.2 presents the choice of which sleep state is the most suitable for the unused hosts, depending on the usage rate of the cluster. The strategy proposed can handle individually each of the states presented or manipulate the Best-fit Energy-Aware Algorithm. We can classify the states into sleep mode levels of intrusiveness, the least intrusive that keeps the host with only some minimal components connected to the total shutdown of the host. This classification is based on the behavior of the states presented in Section 4.1. This means that more intrusive states can save more power when there is a low usage rate of the cluster. However, when the usage rate of the cluster increases, the time incurred to change the idle state to one of the sleep states, plus the power consumption associated with this event, do not allow its use. Therefore, in clusters with a high usage rate, less intrusive states are preferred. We can classify intrusiveness in the following order, from least intrusive to most intrusive: standby, hibernate and poweroff.

Algorithm 4.2 Power state selection procedure

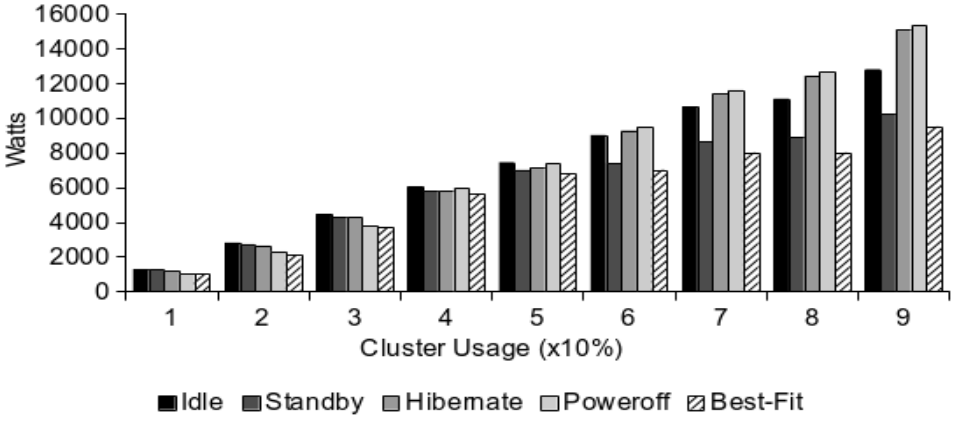
```
for ∀hosts ∈ Managed physical machines do
    allhostsList ← getAllhosts()
    freehostsList ← getFreehosts()
    for ∀freehost ∈ freehostsList do
        if allhostsList - freehostList <= 30% then
            apply poweroff to freehost
        end if
        if allhostsList - freehostList < 50% then
            apply hibernate to freehost
        end if
        if allhostsList - freehostList >= 50% then
            apply standby to freehost
        end if
    end for
```

Thus, the Best-fit Energy-Aware Algorithm uses thresholds of the cluster use, to decide the best strategy that will be applied to each new time, depending on the cluster usage rate (by default, for each new job submitted). These thresholds can be seen in Table 4.1. These percentages are based on the primary experiments presented in Section 4.1, which showed the way a state saves more energy, depending on the cluster usage. Algorithm 2 checks the total number of hosts (*allhostsList*), and the amount of free hosts (*freehostsList*). Using these values, the rate of use of the cluster can be verified on-the-fly. Based on this information, the different sleep states can be applied to unused hosts. We can see that the relationship between power consumption and turnaround time of jobs within the thresholds shown in the table correspond to intervals with the highest usage rate for each strategy.

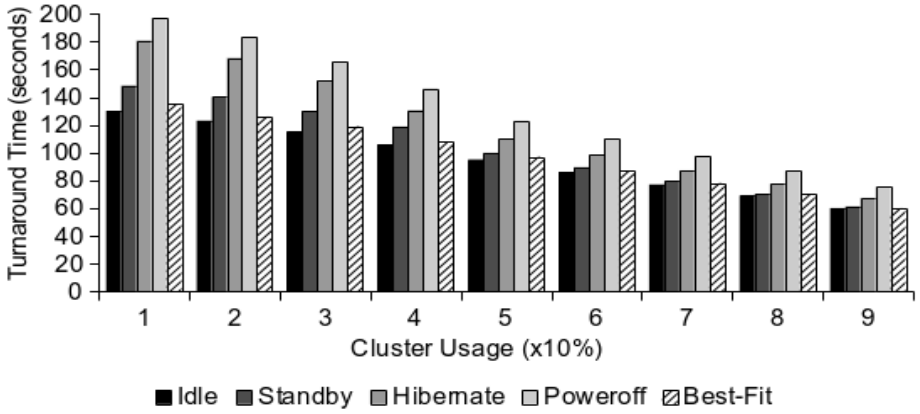
These choices are based on the impact of each sleep state on the turnaround time. The greater the power savings (G2—poweroff), the greater the impact on turnaround time due to the time required for the host is active again. With a low rate of use, this is a suitable technique, because the cluster retains a lot of idle hosts the majority of the time. On the other hand, in a cluster with a high rate of usage, a technique that allows a rapid return of the hosts to the ready state is a deciding factor on turnaround time.

4.1.3 Experiments and Discussion

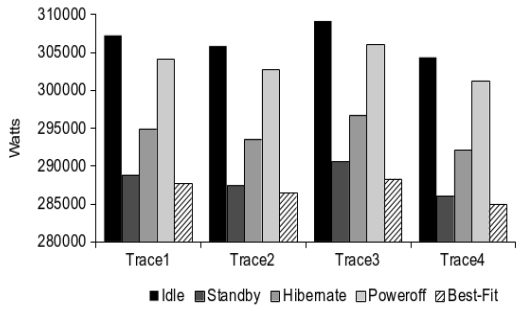
When applying our power strategy on the previous scenario showed in Section 4.1, we can see in Figure 4.3(a) that the results become more expressive in both power-savings and job turnaround time. Thus, when the job runtime ends, the host is then released and becomes available for the next job. It is not directly placed into one of the sleep states. The strategy checks in the job queue if there are workloads that are waiting for resources. In this case, there is no time loss or power consumption increased, to put the host in a sleep state



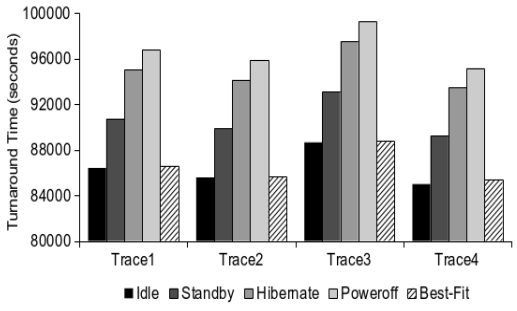
(a)



(b)



(c)



(d)

Figure 4.3 – Best-fit Energy-Aware Algorithm Evaluations

and make it return to a ready state. Therefore, this strategy provides benefits for all states used. For states that are slightly intrusive, power-saving is increased, and for the states that are quite intrusive, allowing power savings because it has fewer changes between states. Figure 4.3(b) shows the impact of these changes on the job turnaround time, and we note that in all situations that impact on users jobs time decreases because in this scenario there are fewer context switches between the 'ready to run' state and sleep states. This is also

due to the behavior of jobs under the states, because in this scenario, there is less time to be added to the job turnaround time, to shut down and restart hosts.

Although all states have taken advantage of the resources allocation, the best usage of this allocation was the Best-fit Energy-Aware Algorithm, which achieved the highest power-savings in all cases. As observed in Figure 4.3(a), the proposed strategy showed more power-saving than the poweroff while the cluster was under low usage. This happens due to the dynamicity of the proposed strategy, which always chooses the least intrusive state. When we used our strategy about the scenario of real workloads, we obtained satisfactory results. In Figure 4.3(c), states that show less intrusiveness have been able to increase power savings, and the states of greatest intrusiveness changed behavior that wasted power to power-saving too. This shows that the Best-fit Energy-Aware Algorithm proposed in this work can save power, using the sleep states wisely. As for the turnaround time, the biggest difference can be seen in Figure 4.3(d). Besides allowing less intrusive states, impacting even less execution time of jobs, it has enabled the turnaround time of more intrusive states to fall by half. This shows that our proposed strategy performs its main goal, saving power while decreasing the impact on the turnaround time of jobs. The behavior of the Best-fit Energy-Aware Algorithm, when we use a real trace, presents energy savings and a great correlation with the job turnaround time. We can see in Figure 4.3(c) that the four tests with the SDSC trace, our strategy presented the best energy saving. We can see an average savings of 19% on the idle state with and 12% more turnaround time.

Results showed that the best state to save energy is in the standby mode. This state allows all hardware components (less memory) to be turned off and allow a quick return to a ready state for a new execution. Hibernate and poweroff are more intrusive states completely turning off the host, leaving only a small hardware implementation checking out a warning to its new restarting. This context switches between turning off and restarting hosts takes much time and power, so many context switches take precedence over the benefits on these power-saving states. The proposed strategy shows a greater influence precisely in these last two states since it can make these states save energy even with a high usage rate of the cluster. The turnaround time, a critical factor for the user of an HPC cluster, was also affected by the use of the amount of states, which allowed keeping a low impact at this time. Moreover, the Best-fit Energy-Aware Algorithm proposed balances the power consumption of the cluster as a better strategy can adjust as the usage rate of the cluster. This provides a solution which changes the power consumption characteristics of the workload dynamically.

4.2 Concluding Remarks

In this chapter, we investigated the impact of power-saving techniques on job turnaround time in HPC clusters, since such techniques might impact the job execution times while alternating the hosts among different sleep states and even turn hosts off for some period.

Based on these results, we propose a new strategy that switches from one state to another depending on the workload and on the RMS's job queue. Experiments with an enhanced version of the SimGrid simulation tool showed results up to 55% of energy savings with synthetic workloads and 19% with real workloads.

As expected, these savings are strongly related to the cluster usage rate. With a light workload, cluster hosts remain idle for longer periods of time so that switching to more economic states bring more energy savings even with the high overhead to enter them.

The aim of this chapter was answering research question 3, using the knowledge acquired in the evaluations of the previous chapter, and expanded them into a multi-host environment interconnected by a network. The results showed that the choice of techniques and mechanisms appropriate to the behavior of jobs directly impacts on the application performance. For cluster environments, the reduction in the processor frequency is not attractive, because this operation reduces the number of instructions that the processor can perform in a given interval time. However, the cluster jobs behavior is different from the clouds services behavior. We strongly believe that with this chapter we contributed to a better understanding of the trade-offs involved in power-saving states on an inter-host environment. The next chapter we discuss these differences and assess the impact of techniques and mechanisms already studied in that new context.

5. ENERGY-EFFICIENT TECHNIQUES AND MECHANISMS ANALISYS ON CLOUD ENVIRONMENTS

The previous chapter evaluates the trade-off between performance and energy saving in a cluster. The behavior of jobs and tasks running on a cluster is characterized by a finite time for their executions. This runtime should be smaller, and for that, the jobs tend to use the greatest amount of available resources.

Unlike clusters, cloud environments do not have jobs, but services. The service has no finite time, and they must always be up and available for customers to connect. Moreover, the load imposed by a service is floating, i.e. most of the time it does not use all available resources. When the number of customers increases, more resources are allocated to maintain SLAs. When the number of customers decreases, resources are released to save costs. Thus, it offers opportunities to reduce IT costs by computing power with high scalability it provides, or the chance to generate new lines of business possible. Still, many companies now begin combining their local resources with public cloud services, creating the called hybrid clouds, further expanding the impact of this new computing approach.

Based on that, this chapter analyzes the trade-off between energy savings and application performance, taking into account this new approach. First, we performed an evaluation of the impact of DVFS in the virtual data center, given performance. This result has shown that the network can be a determining factor in the decision to improve the trade-off analysis. Thus, we conducted a new experiment showing that depending on the physical substrate, data center performance and energy savings can be achieved with the choice of particular cloud scenarios.

5.1 DVFS Impact on virtual networks performance

Cloud computing aims at providing scalable and on-demand IT resources (e.g., processing, storage, database) through the Internet. These resources can be accessed from anywhere, anytime, using any computing device, such as desktops, tablets or smartphones. The market movement towards cloud computing and IT services outsourcing favors the business of data centers, but the segment still faces major challenges, particularly regarding capital expenses and power consumption costs.

According to a report from Stanford University [Mon12], power consumption in data centers has increased significantly in the last years. Between 2005 and 2010, energy consumption increased by 56% around the world (36% only in the United States). Beyond economics, energy consumption affects other issues such as cooling and emission of harmful gasses.

To enable energy savings, new proposals have been presented from green data center designs, using natural air cooling, to the use of particular technologies that optimize resources utilization. Virtualization [NS07, ZZA10, HR10] is one of these technologies, which serves as the core infrastructure of current cloud computing environments, and due to its features such as virtual machines migration and server consolidation, enables the reduction in energy consumption. Also, there are also technologies that allow energy savings in data center servers, putting servers in standby or altering processing performance to adequate workloads demand, and consequently decreasing energy consumption.

In particular, Dynamic Frequency and Voltage Scaling (DVFS) [SKKE11, HKG+07] is a technique frequently used to save energy on servers. DVFS is especially interesting in data centers that employ virtualization, where each server hosts a different group of virtual machines with diverse aggregate resources demands. However, recent studies [Mat07, BJL+06] show that changes in processor frequency can directly impact on the performance of network-dependent applications. This can be a decisive factor for the utilization of DVFS in data centers that support cloud services, since when the processor frequency is reduced, the processing capacity of the host is compromised, affecting all other components, including the network.

Based on the above observations, there is a trade-off between using DVFS to save energy and network performance, which can directly impact on applications' QoS and SLAs. This section aims to verify the impact of DVFS on network intensive applications performance running on a virtualized infrastructure (Citrix XenServer). The experiments were performed using three different DVFS governors, covering all possible configurations of processor frequencies allowed. The experiments were performed using a synthetic benchmark simulating a web application, in which an external client performs multiple requests through the network.

5.1.1 Experiments and Discussion

This section presents the experiments performed, describing the testbed, benchmarks, DVFS settings and network metrics used. Afterward, the results obtained are presented and analyzed.

Testbed

Evaluations were performed on a client-server architecture, simulating a client host accessing to virtualized applications in a server host, connected by a Gigabit Ethernet network. The server used in our experiments consists of 2 Intel Xeon E5520 (16 cores in total),

2.27GHz, 16 Gb RAM. This server runs the Citrix XenServer, a well-known virtualization solution in the industry.

In each set of tests, DVFS was configured with three operating governors: *performance*, *ondemand* and *powersave*. The energy consumption was obtained using a multimeter which is connected between the power source and the server. This device (EZ-735 digital multimeter) has a USB connection that allows periodic external reading and gives the values of power consumption in watts-per-hour.

The network performance metrics evaluated during the experiments were: throughput and jitter. Throughput is the value that indicates the effective data rate transfer per second, while jitter is the variation in delivery time of packets in a given space of time. This variation is directly related to the network demand. The evaluation of throughput focused on the impact in energy savings and response time to the user. The assessment of jitter aimed at analyzing the impact of the virtualization layer in the variation of data packets delivery, which consequently impacts on energy waste.

The experiment architecture is described in Figure 5.1. The client part of the benchmark performs requests, using the network, to applications hosted on two distinct virtual machines. Each virtual machine is associated with one of the two processors available, forcing that changes in the frequency of both processors can directly influence each virtual machine, and consequently, the application within each one of them.

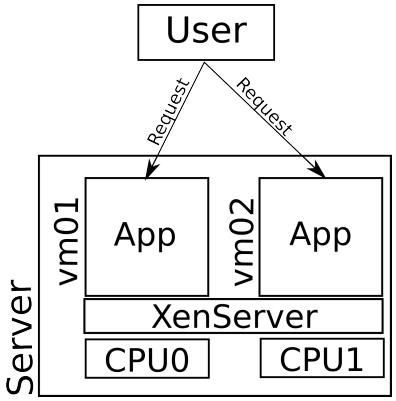


Figure 5.1 – Experiment Architecture

Evaluations of the trade-off between the impact of changes in processor frequency and network throughput were evaluated and monitored through the benchmarks: Hping [San15], T50 [Bri15], Apache-Bench [Fou15] and Iperf [NLA15].

The first benchmark used was Hping. This benchmark is a packet generator that is used to analyze TCP/IP protocols. Currently, in its 3rd version, hping is one of the standard tools for security auditing and testing of firewalls and networks. Hping is programmable using the Tcl language which allows programmers to develop their scripts for manipulation and analysis packages.

The second benchmark used was T50 Sukhoi PAK FA Mixed Packet Injector. This tool was developed for the purpose of packet injection, designed primarily to test DoS/DDoS attacks. From the basic use of stress testing, T50 is capable of sending requests as follows: a value higher than one million packets per second of SYN Flood (+50% of the uplink network) to a network 1000BASE-T (Gigabit Ethernet) and more than 120.000 packets per second of SYN Flood (+60% of the network uplink) in a 100BASE-TX (Fast Ethernet). Additionally, it can send ICMP , IGMP, TCP, and UDP protocols sequentially (with only microseconds difference). It is licensed under the GPL version 2.0.

The third benchmark used was Apache-Bench. This benchmark can measure the HTTP server performance, running concurrent requests, and is especially efficient for test environments where Apache runs on multicore. The metric to be evaluated consists of requests per second at a given time interval, allowing to visualize the impact of various hardware components on web server performance.

The last benchmark used was Iperf. This benchmark is used to test various network metrics such as bandwidth and jitter, which can perform packet injection (TCP and UDP) to measure the performance of these networks. This tool was developed by DAST (Distributed Applications Support Team) and the NLANR (National Laboratory for Applied Network Research), and it can run on many platforms, including Linux, Unix, and Windows.

The first evaluation is shown in Figure 5.2 presents the virtualized server performance to answer requests in a given time interval. The results show that performance and ondemand governor kept the 10000 requests, ending its run in a shorter time than powersave which managed to answer on average smaller requests. The ondemand governor takes a little more time to complete its execution when compared to the performance governor, as there is an overhead in setting the frequencies to the behavior of the application. The powersave governor behavior is an expected result because the processor frequency is limited to one lower than the other two governors.

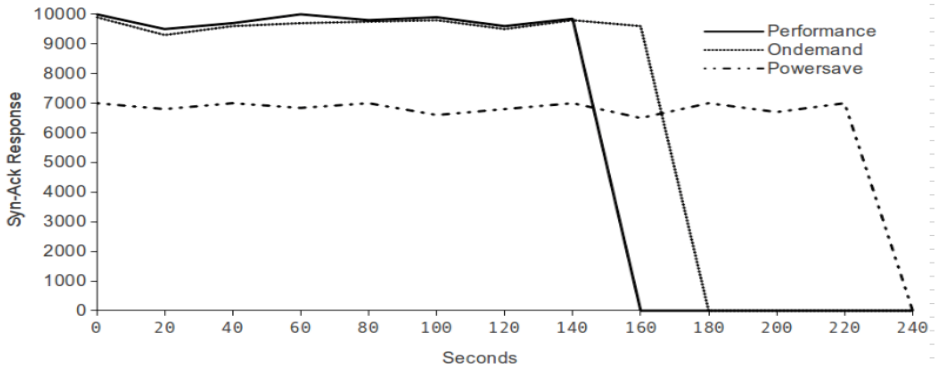


Figure 5.2 – Hping Performance

Figure 5.3 shows that there is little difference in energy consumption between performance and ondemand governors. This happens according to the benchmark behavior,

which always tries to keep the processing to the highest during the test period. Therefore, the frequency variation that enables ondemand governor is quite limited. A big difference could be seen in a case where there is a low rate of requests and, consequently, a low level of processor utilization. However, there is a significant difference between these two DVFS governors and the powersave governor. Despite this governor save around 10% of energy, there is an increase in response time by 70%.

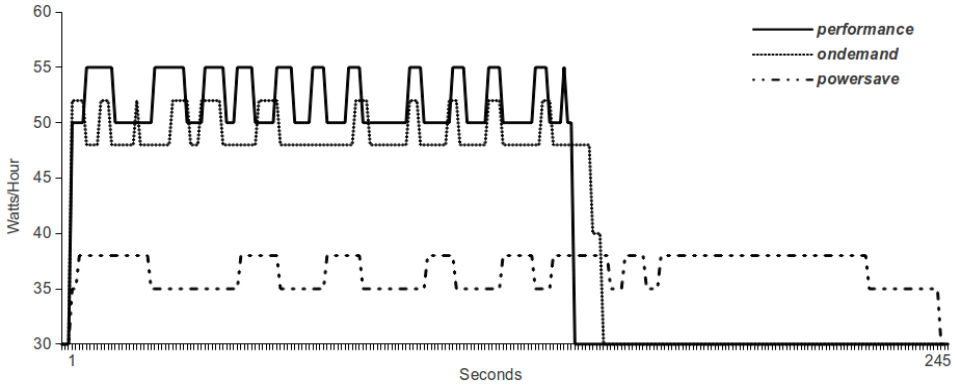


Figure 5.3 – Hping Power Consumption

The second benchmark (T50) tested the performance of the web server again, through a flood of requests, trying to keep for a period of time of time, the most supported requests. Performance and ondemand governors managed to maintain the service in an average time of 150 seconds. Instead, the powersave governor was able to answer only an average between 6000 and 7000 requests over a period of about 68% higher.

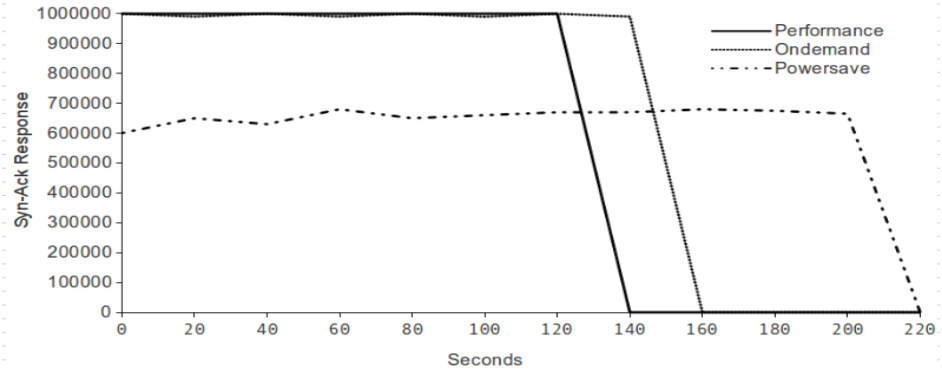


Figure 5.4 – T50 Performance

The T50 benchmark shows similar results in power consumption behavior. These results can be seen in Figure 5.5. Again, there is no significant difference between the performance and ondemand governors. Regarding powersave governor, this enables energy savings of 15% when compared to the performance governor.

Tests using Apache-Bench perform requests to a real HTTP server. In this experiment, were performed a range between 100 and 1000 requests per second, evaluating how many milliseconds would lead the server to respond to all of them. Figure 5.6 shows the

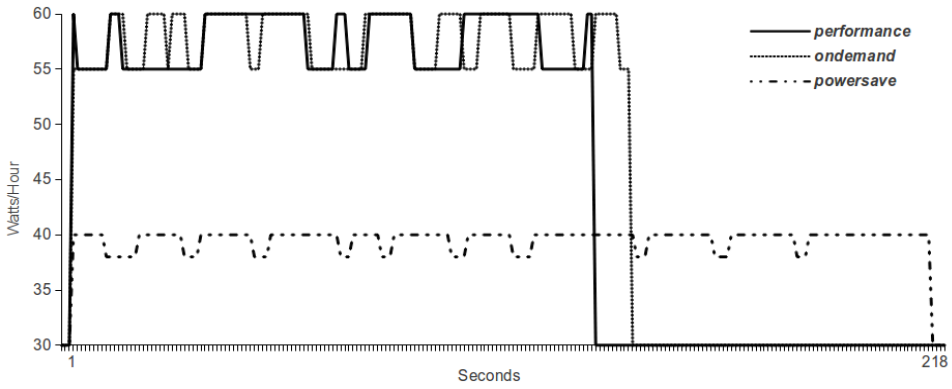


Figure 5.5 – T50 Power Consumption

higher the number of requests, the greater the response time in milliseconds. The ondemand governor is very near to the response times achieved by the performance governor. Both have a response time for all cases on average 35% faster than the powersave governor, which shows that the frequency of the processor directly affects the performance network applications.

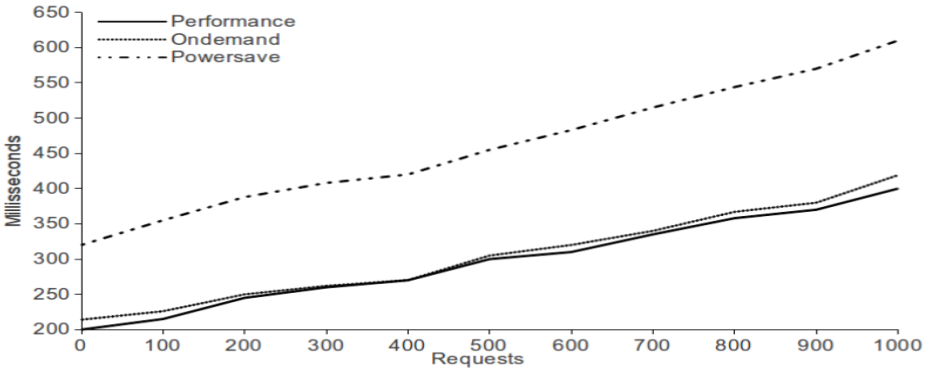


Figure 5.6 – Apache-Bench Performance

Concerning power consumption, Figure 5.7 shows that performance and ondemand governors try to keep the highest processor utilization during the execution time of the application, to respond to the requests in the shortest time possible. With the limited frequency of the processor in powersave governor, there is much energy-saving, although its impact is significant on performance.

Figure 5.8 shows the jitter test. In these tests, DVFS governors from a native Linux environment were compared to virtualized DVFS governors. The results showed that there are differences when comparing jitter on the environment in any of the native DVFS governors against a virtualized environment. Based on this, it can be verified that virtualized environments cause jitter overhead, which can cause an inefficient service for certain types of applications, such as video streaming. Furthermore, there is also a greater impact when using the powersave governor. This is probably due to the delay imposed by the structure of the ring buffer from Xen.

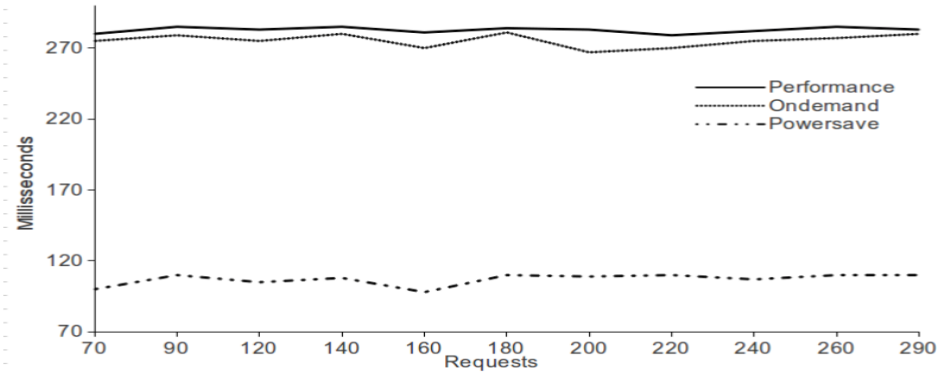


Figure 5.7 – Apache-Bench Power Consumption

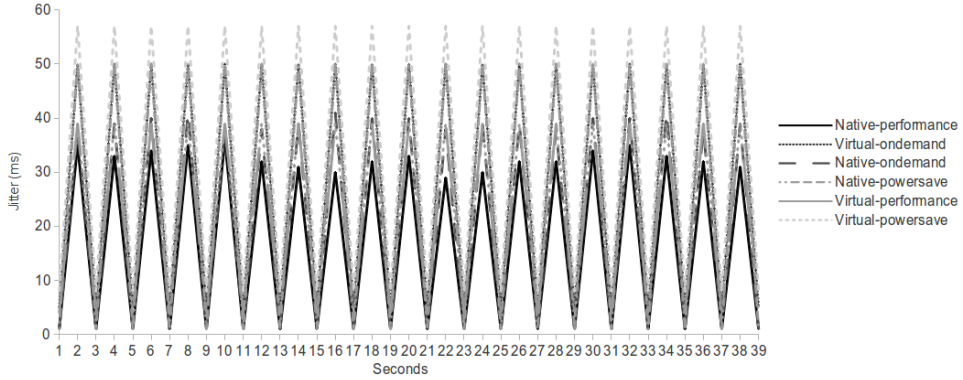


Figure 5.8 – Iperf Jitter Evaluation

The evaluations performed allowed an examination of issues of Quality of Service (QoS) for virtualized networks. The QoS is defined regarding the Service Level Agreements (SLA) with features such as the least throughput, maximum response time or latency time. A network architecture that can manage traffic dynamically according to SLAs is not only essential for the future competitiveness, but can also set the basis for a systematic approach to energy efficiency. However, the implementation of QoS can increase the total network traffic and energy consumption of their virtualized environments.

The tests showed that by increasing the bandwidth, latency increases. For energy consumption, it is necessary to improve the latency by increasing the bandwidth, which directly impacts on energy consumption. On this point, it must be dealt aspects such as component choice and consolidation of I/O. Likewise, it is necessary to investigate networks without loss in performance compared to the bandwidth and energy efficiency. For example, package lossless network protocols usually mean more complex and more latency, as well as more processing power and low-bandwidth efficiency.

5.2 Server consolidation vs DVFS

One of the biggest challenges faced by current data centers is how to reduce their enormous energy expenditure [KL14]. High energy consumption (EC) leads to heat dissipation, which brings with it the need for more powerful cooling equipment, and consequently increases the emission of harmful greenhouse gasses into the atmosphere. The reduction of this impact is the focus of extensive amount of current research. However, when techniques to reduce energy consumption are used, the performance of applications running in this environment is also reduced.

Several studies have modeled answers to this problem by using many techniques and mechanisms that can somehow reduce the energy consumption in data centers. Our work focuses on the two most widely used: the server consolidation via virtual machine migration and the reduction of the processor frequency.

Server consolidation via Virtual Machine (VM) migration has become an important technique to improve the energy efficiency of data centers. Most approaches to energy saving in virtualized environments are aimed at the organization of virtual machines on the available infrastructure, considering various factors such as the energy consumption of physical hosts, the flow of communication between these hosts, and the workload behavior, among others [BAB12]. This organization is not a trivial task, as this decision should take into consideration issues such as performance, relationships between the use of the processor and memory, service level agreements (SLA), and quality of service (QoS). Thus, the possibility of virtual machine migration coupled with a the proper management about the communication channel between hosts and the location of these VMs on the infrastructure became the main factors of the use of consolidation aiming energy-savings [BAB12].

In addition to server consolidation via VM migration, new processor architectures allow the change of its frequency at runtime. One of such techniques is called Dynamic Voltage and Frequency Scaling (DVFS) [EE11]. This mechanism is controlled by the operating system and can be configured from several energy-saving governors that can be adjusted by the user depending on the workload profile.

Workloads that execute on these environments present a significant variation in resource utilization over time. Moreover, there are times when a large number of computing resources are in an idle state or are under-utilized. Thus, both server consolidation and reduction of processor frequency can present opportunities for energy savings. However, both also present penalties on the performance of applications due to the overhead in virtualized environments supporting them.

Thus, the problem evaluated in this paper consists in the determination of which of these proposals should be employed, at what time, and what factors cause more impact on that decision. The hypothesis discussed in this section states that the decision on which

power saving technique or mechanism is the most efficient should be based on the network topology of the cloud environment. This is due to the way the communication between the hosts occurs, and how VM images are distributed to servers. Therefore, it may be suggested that cloud environments with centralized storage are more suitable to support VM migration, while cloud environments with distributed images on the processing hosts should avoid migration and preferably make use of DVFS.

Figure 5.9 shows, at a high level, the discussed approaches for energy savings in virtualized environments. Figure 5.9(a) shows one energy-agnostic environment, where there are two physical machines (PM) underused, but with the CPU frequency configured in the Performance governor.

The result of utilization of server consolidation is depicted in Figure 5.9(b), where the VMs are migrated from the second PM to the first, and the idle PM is turned off. This solution aims at energy saving for better resource utilization of the first PM, at the same time-saving energy through the second PM shutdown.

The result of utilization of DVFS to reduce the processors frequency in underutilized PMs is shown in Figure 5.9(c), where virtual machines are kept on the same PM that is running, but the frequency of its processor is lowered according to the behavior of application running on VMs.

These are the most common scenarios for the use of energy-saving opportunities on data centers. The big question raised by this work consists in which one to use, and in what situation each one is best suited with minimal impact on QoS of applications.

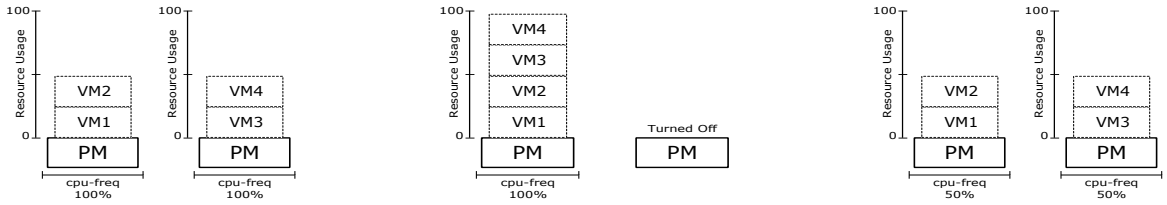


Figure 5.9 – Energy-Agnostic Scenario (a: Two underutilized PMs using 100% of processor frequency) and Two Energy approaches for energy savings in cloud data centers (b: Server Consolidation, turning off the idle PM) (c: Reduction of the frequency of both PMs).

The hypothesis investigated in this paper is that the decision about which opportunity, between VM migration and DVFS optimization, saves energy with the least impact regarding QoS should be based on the network topology and the existence or not of a centralized storage equipment. We believe that the size of the VM is not as relevant as whether the infrastructure features an image repository of centralized VM, or if the VMs are stored in its entirety in allocated PMs. In the case of the existence of a repository (Figure 5.10(a)), it is likely that the best decision is migrating VMs because network traffic between PMs during the migration would be mostly memory pages. However, if there is no centralized repository, in the case of server consolidation, the full images of the VMs should move through the

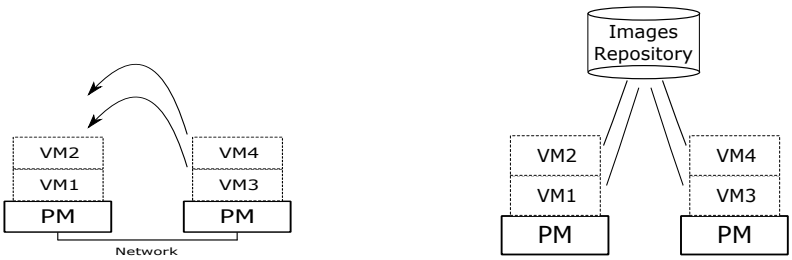


Figure 5.10 – Evaluated in the section (a: VM migration through point-to-point network) and (b: VM migration through centralized images storage).

network (Figure 5.10(b)), which would result in a massive overhead of time. In this case, the best option would be to reduce the frequency of the processor. Tests were conducted to validate this hypothesis.

5.2.1 Experiments and Discussion

The tests were conducted between two hosts equipped with Intel Xeon processors E5-4650 v2 2.40 GHz, 40/80 cores, 25 MB L3 Cache. The servers run the Citrix XenServer 6.2, a well-known virtualization solution in the industry. Between these two hosts, we tested two different types of networks: Gigabit Ethernet [Sei98] and Infiniband [Sha02]. We used these two categories of networks with the various speeds to verifying the impact to the size of the virtual machines on networks with different flow rates, given the fact that Infiniband networks can be up to 3 times faster than the Gigabit Ethernet, offering a negligible power consumption (PC) [BCL+11].

In tests where we performed access to shared storage, we used the PowerVault MD3200i SAN Storage Array, configured with 1 Gigabyte iSCSI [SMS+04] as a communication protocol between storage and the host. To range the size of VMs, we used as a basis the VM sizes offered by Amazon Web Services, as described in Table 5.1.

Table 5.1 – VM size based on Amazon EC2.

Type	*CPUs*	*Memory*	*Image Disk*
Tiny	1 CPU	1 Gb RAM	1 Gb
Small	1 CPU	2 Gb RAM	4 Gb
Medium	2 CPUs	4 Gb RAM	8 Gb
Large	2 CPUs	8 Gb RAM	16 Gb

As the application load on each VM tested, we used a stress test. The energy consumption was obtained using two power meters connected to the source of energy and the servers. These devices (Accuracy: $\pm$0.3%+2D) have USB connection that allows periodic external reading and gives the values of power consumption in watts.

All tests were carried out taking into account the two scenarios proposed in this paper, in which initially there are two VMs over two PMs. In the first scenario, one of the VMs is migrated in another PM, making the idle PM be turned off to save energy. This scenario can be divided into more sub-scenarios, depending on the location of the images that support the VMs: on the PMs or in a centralized storage. Another factor tested was the communication between the hosts and the storage via iSCSI, or if there is no centralized storage, the communication between point-to-point hosts using Gigabit Ethernet or Infiniband. In the second scenario, VMs remains on your PMs just by changing the frequency of the processor of each PM.

The power consumption measurements for each scenario were performed at two different times. Figure 5.11 shows the behavior of the majority of the tests and supports the understanding of the results presented in Tables 5.2 and 5.3. In this Figure, two PMs, running one VM each, start their execution in t=3s. As both PMs are underutilized, between t=9s and t=12s, the process of consolidation is triggered, where the VM allocated in PM2 migrates to the PM1. Consequently, the PM1 increases its use of resources now with 2 VMs, and the PM2 becomes idle and is turned off. Thus, in t=12s the power consumption is measured only in PM1.

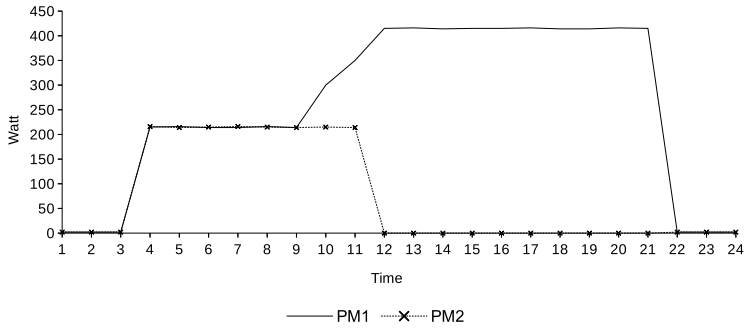


Figure 5.11 – Example of VM consolidation.

Table 5.1 shows measurements performed during the VM consolidation (in Figure 5.11, the interval between t=9s and t=12s). The tests were conducted in four separated sets: the first set (tests 1-4) consists of VM consolidation through a shared storage via iSCSI protocol; the second set (5-8) consists of VM consolidation through point-to-point Gigabit Ethernet; the third set (9-12) consists of VM consolidation through an Infiniband network; and the last set (13-16), consists in reduction of processor frequency (adjusted to the behavior of workloads) and consolidation is not utilized.

In the tests, we used four different sizes of VMs as mentioned above, for each test set. The table shows the total time the VM consolidation led to being completed for each case, and the last set demonstrates the time of the exchange of DVFS states (assuming one second for each of them).

Table 5.2 – Performance and energy-efficiency evaluation of each evaluated VM storage alternative - A: VM migration through the storage, B: VM migration through the network, C: DVFS on 2 hosts without VM migration.

Test	*Proposed Arch.*	*Comm. Protocol*	*VM Size*	*Total Time(s)*	*host 1 EC(kWh)*	*host 2 EC(kWh)*	*Storage EC(kWh)*	*Total EC(kWh)*	*EC(kWh) over VM Consolidation*
1			Tiny	8	0.215	0.215	0.20	0.63	0.0014
2			Small	10	0.24	0.24	0.21	0.69	0.0019
3	A	iSCSI	Medium	12	0.275	0.275	0.216	0.766	0.0025
4			Large	15	0.305	0.305	0.22	0.83	0.0034
5			Tiny	240	0.325	0.325	-	0.65	0.0433
6		Gigabit	Small	490	0.345	0.345	-	0.69	0.0939
7		Ethernet	Medium	925	0.36	0.36	-	0.72	0.185
8	B		Large	1800	0.37	0.37	-	0.74	0.369
9			Tiny	50	0.3	0.3	-	0.6	0.0083
10			Small	150	0.34	0.34	-	0.68	0.028
11		Infiniband	Medium	340	0.37	0.37	-	0.74	0.0698
12			Large	740	0.374	0.374	-	0.748	0.153
13			Tiny	1	0.209	0.209	-	0.418	-
14			Small	1	0.238	0.238	-	0.476	-
15	C	-	Medium	1	0.269	0.269	-	0.538	-
16			Large	1	0.295	0.295	-	0.59	-

In an analysis of the first three sets (where VM consolidation is applied), we can see that consolidation through a point-to-point network over Gigabit Ethernet causes higher energy consumption when we take into account the time needed to carry out the migration of the VM. Although the final use of the tests with Infiniband network show increased energy consumption, it takes almost a third of the time to complete the migration process, when compared to Gigabit Ethernet. However, the best design aimed at energy savings in VM consolidation is the one that uses a centralized storage.

This can be explained by the different characteristics of VM migration in a point-to-point or via shared storage. When VM migration occurs in a point-to-point network, all the VM image is copied over the network, from one PM to another. This causes an overload on the network throughout the process, and even on fast networks such as Infiniband, depending on the size of the VM, the migration process can take a long time. This overload does not exist when the images are located on a shared storage because in this case only a reference to the image within the storage is created in the destination PM, and network traffic between the two PMs is used most of the times only to update memory pages.

Therefore, we can define the design that further improves the trade-off between performance and energy saving is in a VM image storage, even with the increase of power consumption of the storage, where there is a shared storage. However, the results of using DVFS also seem to be very promising, considering that they do not impute migration time and presents a total low energy consumption.

As the DVFS does not carry out migration, we must compare the energy consumption of using DVFS in both PMs or the option to migrate using shared storage, in the moments after the migration (as we can see in Figure 5.11, between t=13s and t=22s). This is the time when the idle PM is turned off, and the two VMs are running on a single PM, allowing max-

Table 5.3 – Energy consumption after VM consolidation or DVFS for different image sizes.

VM Size	*VM Consolidation through the storage*	*DVFS on 2 hosts without VM Consolidation*
Tiny	0.415 kWh	0.418 kWh
Small	0.44 kWh	0.476 kWh
Medium	0.475 kWh	0.538 kWh
Large	0.505 kWh	0.59 kWh

imum energy savings. The results of this evaluation can be seen in Table 5.2, where VM Consolidation option is also considering the power consumption in the storage.

Results show that, regardless of the size of the VM, when we have the option to keep the VMs on a shared storage, this is the best choice to save power. On the other hand, if the data center design does not present a shared storage, or if the PMs are interconnected by point-to-point, regardless of the speed of the network, the best option is to use DVFS.

5.3 Concluding Remarks

In February 2007, the main leaders of the IT industry had announced The Green Grid, a nonprofit consortium whose mission is to improve the energy efficiency of data centers and business ecosystems based on computing. The strategy is to encourage the development of chips, servers, networks and other solutions that consume energy more efficiently.

Some of these efforts have focused on technologies such as virtualization. However, virtualization technology incurs in a processing overhead, through the addition of an abstraction layer that translates all requests between the virtual machine and physical host. This layer is affected by other technologies that attempt to promote energy-savings, such as DVFS.

This chapter evaluated the impact of DVFS on network-dependent applications in virtualized environments, focusing on network performance. The choice of this metric is justified by the impact on response time for user applications. Furthermore, we also evaluated the overhead of the virtualization layer on jitter, a metric that can impact on energy waste, as well as the quality of service.

Besides, one of the greatest challenges in today's virtualized environments management relates to their energy consumption. High energy consumption rates have a direct impact on sustainability issues such as heat generation and its cost with cooling, besides the emission of harmful greenhouse gasses in the environment. Also, these infrastructures can keep many idle hosts most of the time, which would be a possibility for energy saving. Focusing on these challenges, several studies have proposed to use two energy-saving options: DVFS and VM consolidation.

While such technique or mechanism allow energy savings, their simultaneous use can overload the environment with constant changes of states, impairing application performance. As cloud environments are heterogeneous, there must be a beacon that allows inferring the best technique to be used, taking into account the specificity of each environment. The choice of the best technique to use is strongly linked to the existing network infrastructure in the physical substrate. This is due to the ability of virtualized environments to support VM migration among available resources. This VM migration can be reasonably costly, and the manner in which the network is set becomes the deciding factor on the suitability of this technique.

Our hypothesis that the choice between migration or reducing the processors frequency necessarily passes through the method employed for network topology and the storage of VM images. In architectures where the VM images are stored in a repository and when migration occurs, only memory pages are transferred through the network. Thus, we believed that this scenario offers a suitable case for performing VM consolidation. On the other hand, when VM images are stored on the same host where they are instantiated, migration becomes costly, because, besides the memory pages, VM image files must also be transferred through the network. In this situation, we believed that the most energy-efficient option would be using DVFS to reduce the processor's frequency.

The aim of this chapter was answering the research question 4, differentiating the tests previously performed about cluster jobs and introducing new tests focused on clouds service behavior. To evaluate this hypothesis, experiments were conducted with several VM sizes using different network technologies and the two power-saving options discussed above. The results showed that the virtual machine images location really should influence the decision between using DVFS to reduce the processors frequency or migrate VMs to consolidate resources. The results presented by the evaluations of this chapter, added to the test results presented in the previous chapters, are the basis for the strategy proposed in the next chapter, which uses all the knowledge acquired in the evaluations carried out so far, to choose which power-saving states and mechanisms best suited to the cloud data center environments, providing the lowest impact on performance.

6. ENERGY-EFFICIENT CLOUD ORCHESTRATOR (E-ECO)

There is a worldwide effort when it comes to reducing energy consumption, either by the choice of renewable energy generation technologies or the use of solutions to reduce energy consumption by manipulating the options available for each organization type. In particular for large-scale computing environments such as cloud data centers, virtualization itself is a platform that enables energy saving, by better use of resources. Furthermore, hardware and software offer other technologies that allow manipulation by the user-level toward the same goal.

Energy saving options are not widely used in such environments because the energy saving options impact on application performance. The reduction in the processor frequency reduces the number of instructions performed, increasing the execution time of processes. Virtualization also causes overhead over execution times due to the time between migrations or transactions through the virtualization layer.

Such trade-off between performance and energy saving has been discussed and evaluated in this work, for intra-host environments in Chapter 3, for inter-host environments in Chapter 4, and for virtualized data center environments in Chapter 5. Based on the findings of these previous evaluations, in this chapter we present a prototype to balance better the overhead imposed by energy saving options with application performance.

The description of the approach, called e-eco, is organized following a typical software development process [Som01]: requirements, specification, architecture, implementation, deployment, and evaluation.

6.1 Motivating Scenario

Most studies in the cloud computing area focus on only one metric, namely performance, without considering power trade-offs. This decision is based on several reasons: performance directly impacts the quality of user experience, while power consumption is an operating cost that, although ends up transferred to the end customer, does not reflect on the quality of experience. Moreover, energy consumption does not violate predefined SLAs. This mindset on the part of cloud providers began to change with the decision to reduce taxes for large data centers, insofar as they reduce energy consumption. Added to this, the new generation of supercomputers aiming to achieve exascale processing have limits regarding energy consumption. This motivated many studies in the last few years aiming to improve the trade-off between performance and energy savings for large-scale environments.

Our work targets this context. E-eco is an orchestrator of infrastructure capabilities in private cloud environments, where scalability is determined by adding/removing virtual

machines, with the intention of meeting a demand of requests from customers. Such work is necessary because several cloud data centers providers maintain a number of physical servers idle or underutilized. The speed explains such a practice that these hosts can respond to a sudden growth in demand from service customers, and energy-saving options can penalize this speed in the hosts' response. However, there is an opportunity to save energy on these hosts, causing little impact on response time or other performance metrics. Besides, some energy decisions using reduced frequency of processors in conjunction with server consolidation are misleading, because if there is a free slice of resources in a consolidated server where the DVFS can act, that server can get more virtual machines without reducing the processor frequency. Besides, sets of virtual machines with different frequencies at the same host is only possible in a few processor architectures.

6.2 System requirements

The main goal of our orchestrator is improving the trade-off between energy savings and performance applications. In cloud environments, performance can be defined as the speed at which the provider meets a customer request. This can be reflected in metrics such as response time and transactions per second, among others. Energy savings refers to the amount of energy that is no longer spent, which would impact in lower operating costs and also in reduced environmental impact. Energetic data center efficiency is measured by the PUE metric (Power Usage Effectiveness).

Considering this, e-eco targets cloud platform systems that can instantiate virtual machines on physical hosts. Such platform must have total access to infrastructure layer components. Thus, the techniques and mechanisms to save energy can be accessed and manipulated, and balanced with the performance metric.

For this purpose, every time there is fluctuation in the pre-defined SLA and therefore the demand for resources, e-eco checks the need for more or fewer hosts in the sleep state to meet this demand quickly. Thus, e-eco maintains hosts running virtual machines, hosts in an intermediate state of suspension that save more energy than idle hosts at the same time they can turn in running state quickly, and turned off hosts. Whenever e-eco needs to request new hosts for execution, these hosts are searched in the intermediate site, and such site, in most cases, is filled with turned off hosts. The number of hosts in each state is based on a model and decided by the e-eco in time slots based on the maximum time of transitions. Our previous evaluations show that this design can save energy with little impact on application performance.

A key requirement in the design of e-eco is, i.e., the capacity to operate without the need to cloud environment administrator intervention. All power-saving decisions taking into account performance metrics are based on messages that come from the infrastruc-

ture, through APIs provided by the platform. For example, a relevant information consists of the location and organization of virtual machines images. E-eco can look for such information that is provided by the platform, and then e-eco can switch via network infrastructure settings, such as enable/disable DVFS and server consolidation. Our previous evaluations showed that the choice between the use of DVFS or server consolidation is the best option to improve the trade-off between energy savings and application performance, unlike studies that propose the use of DVFS and server consolidation on the same hosts.

As the energy saving techniques and mechanisms, we address in this work are widely used in today's computing environments, our orchestrator can operate in environments that range from small companies to large-scale data centers. E-eco was designed to work with Openstack [Pep11], but we believe it can easily be adapted to other cloud platforms.

6.3 System design and Implementation

The e-eco orchestrator is based on the work by zhu et al. [ZLLW12], and it divides hosts in the cloud infrastructure into three groups: Running (ACPI G0), Intermediate (ACPI S3), and Turned Off (ACPI G2). Figure 6.1 shows this arrangement. The Running Site (RS) contains the hosts that are allocated by running instances and/or hosts that are ending their execution and will be released. In the Intermediate Site (IS), there are hosts in suspended state, waiting to meet the demand for new resource allocations. Turned Off Site (TS) is where turned off hosts are kept.

Our assumption when designing e-eco was that decision on a number of hosts to be maintained at the IS has a significant impact on energy savings of the system. As our approach is implemented at the IaaS level, e-eco calculates the number of required hosts in the IS based on two information: the amount of hosts running applications in the RS, and the frequency in which more resources are needed. In this context, e-eco *moves* hosts from TS to IS, and keeps them waiting for new demands. When there is no more demand, hosts can be released and transferred again to TS.

When the cloud allocates resources to meet new demand from customers, hosts are allocated and maintained with instances running on RS. In this place, there are hosts with different rates of resource usage, as well as hosts that are to be released once they finalize their executions. When hosts are overloaded, new hosts should be allocated. This demand is provided by the IS that keeps hosts in the sleep state, i.e., an intermediate state that saves more energy than idle hosts while responding faster than hosts turned off, resulting in energy saving by not performing very deep state transitions.

The demand of application is estimated by the equation below, where U_{ref} is the per-application agreed value of performance expectation, and U_i is the current metric mea-

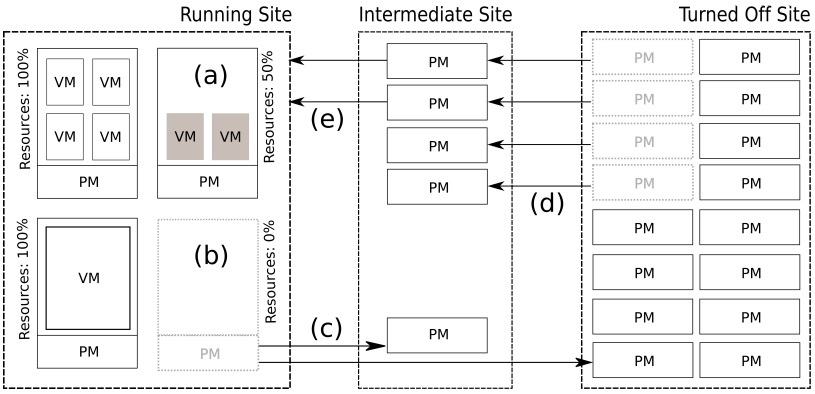


Figure 6.1 – First part of the e-eco strategy. It consists in choosing, based on the number of hosts running and the frequency in which new hosts are required, how many hosts must be placed in an intermediate state, which could quickly meet new demands. This intermediate site increases performance by reducing response time, while the rest of the hosts are shut down to save energy.

surement. High-level metrics such as transactions per second is often measured between the customer and the cloud provider. However, the infrastructure that supports cloud environments (IaaS) typically monitors low-level metrics, such as CPU, memory, and network usage. Translation between high-level metrics to low-level metrics is a very complex challenge in today's cloud environments. Moreover, monitoring high-level metrics are quite difficult, and impact performance and privacy of users. Also, the communication channel between the customer and the cloud listener cannot be monitored by the cloud infrastructure without overhead on the communication channel. For all these reasons, we used the model to estimate U_i.

$$U_i = c_0 + c_1 \cdot avg1 + c_2 \cdot avg5 + c_3 \cdot avg15, \tag{6.1}$$

Where U_i represents the estimated cloud transactions per second of applications in the entire hosts. The coefficients c_0, c_1 and c_2 are the weights assigned to each variable, in each loadavg times. $avg1$, $avg5$, and $avg15$ are monitored available values in `/proc/loadavg`, for 1, 5, and 15 minutes, respectively.

We used this estimated value (U_i) to power the model that calculates the required amount of hosts on IS, to meet the demand of RS.

$$\alpha = 1 - \frac{U_{ref} - U_i}{U_{ref}} - \lambda, \ where\ 0 < \lambda < 1. \tag{6.2}$$

Thus, based on the number of hosts in the RS and its historical, the aggressiveness of λ can be calculated as

$$\lambda = \frac{hosts_{run}}{hostsPrevious_{run}}.\Delta, \ where\ \Delta = 0.5 \tag{6.3}$$

and α is used to derive the number of hosts in the IS. The handling of states of the hosts is performed over the network, using Wakeup-on-LAN (WoL) [WT10] technology, which allows managing different states on the network hosts. This occurs through UDP packets sent directly to the active network adapters of the hosts. These packets are received even by turned (soft) off hosts.

Algorithm 6.1 uses α to decide the required amount of hosts to be added or removed from the IS, intending to keep enough hosts to meet a reasonable demand of applications on RS. As the amount of hosts kept in IS is based on the number of hosts in RS, in case no hosts are running, 10% of turned off hosts are kept in IS (this value can be adjusted). This algorithm seeks the amount of currently existing hosts in the IS from Openstack queue, and based on α, reaches a new value that the IS should have. If there are overuse, the algorithm uses WoL in order to turn on sufficient hosts (β) to complete the necessary amount of hosts in IS. In the case of underutilization, the same technique is used to turn off the hosts from IS.

Algorithm 6.1 Intermediate Site Size Adjustment Procedure

1: **if** $hosts_{run}$ = 0 **then**
2: $hosts_{sleep} \leftarrow hosts_{off}$ x 0.1
3: **else**
4: $hosts_{sleep} \leftarrow hosts_{run}$ x α
5: **end if**
6: **if** $hosts_{sleep} > hostsOld_{sleep}$ **then**
7: $\beta \leftarrow hosts_{sleep} - hostsOld_{sleep}$
8: **for** $\forall\beta$ **do**
9: send *turning on signal* to $PM \in hosts_{off}$
10: send *standby signal* to $\forall PM \in hosts_{sleep}$
11: **end for**
12: **end if**
13: **if** $hosts_{sleep} < hostsOld_{sleep}$ **then**
14: $\beta \leftarrow hostsOld_{sleep} - hosts_{sleep}$
15: **for** $\forall\beta$ **do**
16: send *turning off signal* to $PM \in hostsOld_{sleep}$
17: **end for**
18: **end if**=0

Then, e-eco decides how deallocated hosts from RS should be managed. When a host is deallocated in RS, the algorithm checks whether the α value is satisfied, and if so, turns off these hosts. Otherwise, the algorithm increases hosts in IS. Therefore, Algorithm 6.2 presents a choice based on energy savings and performance. The algorithm can save energy by shutting down hosts at times where there is no need for IS, and it can also bring hosts to IS faster than turning on hosts from Turned Off Site.

When hosts are released (Figure 6.1 (b)), they may be moved to the IS, expecting new resource demands, and avoiding situations where there may be many exchanges either states (Figure 6.1 (b)) or moved to TS.

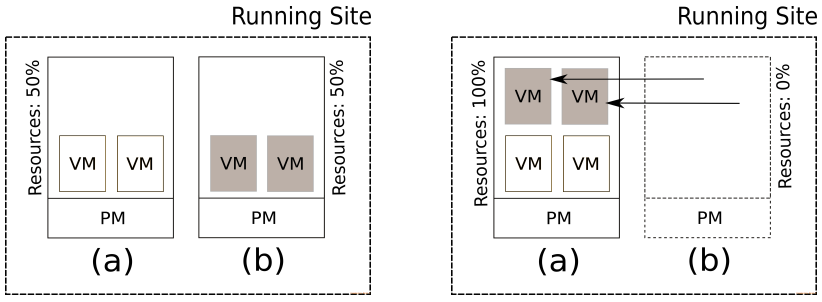
Algorithm 6.2 Running Site PM Deallocation Procedure

1: **if** $\alpha = 0$ **then**
2: send *turning off signal* to $\forall hosts_{freed} \in hosts_{run}$
3: **else**
4: **for** $\forall \beta$ **do**
5: send *standby signal* to $hosts_{freed} \in hosts_{run}$
6: **end for**
7: send *turning off signal* to $\forall hosts_{freed} \in hosts_{run}$
8: **end if**

IS keeps the amount of hosts needed to meet the demand for new resources. When e-eco decides that more resources are needed, new hosts are moved from TS (Figure 6.1 (d)) and, if necessary, new instances are started over those resources and placed on RS (Figure 6.1 (e)). Therefore, IS prevents loss of performance due to the speed to restarting hosts.

TS keeps turning off hosts. When there is a need to add new hosts in the IS, hosts are turned on and kept in a state of suspension (Figure 6.1 (d)). E-eco decides the number of hosts to be moved between TS and IS based on the number of hosts required for IS to meet the demand from RS.

Furthermore, hosts with a low usage rate (Figure 6.1 (a)) present an opportunity for management of target energy savings. Figure 6.2 shows the two possibilities that the proposed strategy allows to be manipulated. The first (Figure 6.2(a)) option is to reduce the frequency of the processors of underused hosts. Another alternative is to consolidate virtual machines from the host (b) at the host (a), allowing host (b) to be turned off or placed in a sleep state, as depicted in Figure 6.2(b).



(a) CPU Frequency Changes (b) VM Consolidation

Figure 6.2 – Second part of the e-eco strategy. It consists in the decision either allowing VM consolidation or reduction of the processors' frequency, depending VM images storage location and network latency.

The choice between these two options necessarily passes through the method employed for storage of VM images. In architectures where VM images are centralized and when migration occurs, only memory pages are transferred through the network. Thus, we believe that this scenario offers a suitable case for performing VM consolidation. On the

other hand, when VM images are stored on the same host where they are instantiated, migration becomes costly, because, besides the memory pages, VM image files must also be transferred through the network. In this situation, we believe that the most energy-efficient option would be using DVFS to reduce the processor's frequency.

Algorithm 6.3 manages the decision on either consolidating VMs or using DVFS. As this decision is based on the existence of a centralized storage for VM images, Algorithm 6.3 checks if there is an available storage in the infrastructure. This information is provided by the OpenStack Image Service API and accessed via Ceilometer. If there is an available storage, e-eco enables VM consolidation and disables the support for DVFS. If there is no centralized storage in infrastructure, VM consolidation is disabled, and the DVFS becomes enabled. The necessary changes on VM consolidation issues are performed using libvirt [BSB+10], an API that provides resource management in virtualized environments. DVFS can be enabled and disabled using Simple Network Management Protocol (SNMP).

Algorithm 6.3 Running Site Energy-Efficient Selection Procedure

```
1: for ∀PM ∈ hosts_run do
2:     if exist 'Storage' then
3:         enable VM Consolidation using libvirt
4:         disable DVFS Support on ∀PM using SNMP
5:     else
6:         enable DVFS Support on ∀PM using SNMP
7:         disable VM Consolidation using libvirt
8:     end if
9: end for
```

In summary, Algorithm 6.1 decides the number of hosts that must be kept in an intermediate condition based on the performance metric and the number of hosts running. Algorithm 6.2 determines what action should be taken when a host is freed. Algorithm 6.3 decides, based on the established architecture, if the environment must perform VM consolidation or offer a reduction in the processors frequency. This set of three algorithms is able to adjust the cloud environment managed by Openstack in a way that it saves energy while maintaining a number of hosts to meet new demands, thereby maintaining compliance with performance metrics.

Nonetheless, the changes of hosts states among the three sites provided by e-eco can not be performed at all times, because α could increase rapidly, causing overload. Therefore, changes applied by e-eco must be carried out in periods where there are greater energy savings, taking into account all factors that can influence that decision.

Thus, we model the energy consumed during the idle time of the hosts denoted by

$$E_k = P_k T_k \tag{6.4}$$

Where P_k is the idle state power consumption, and T_k is the host duration time in the state. To reflect our proposal, we need to add to this model the cost during state transitions regarding power consumption and duration time.

$$E_x = P_k T_{k'} + P_i T_i + P_j T_j + P_n T_n \tag{6.5}$$

P_i and T_i refer to the power consumption and time to go into a sleep state. P_j and T_j relate to the power consumption and time spent in the sleep state. P_n and T_n concern the power consumption and time to exit a sleep state.

Generally, P_j is smaller than P_k, and P_i and P_n are generally larger than P_k. This means that to save energy, T_j must be long enough to offset the increased in energy consumption during T_i and T_n. Based on this, it can be noted that the shortest time interval which is energy efficient occurs when $E_k = E_x$.

Algorithm 6.4 Idle Checking Procedure

```
1: for ∀PM ∈ hosts_run do
2:     if host_idle_time >= E_x then
3:         enable e-eco capabilities
4:     else
5:         disable e-eco capabilities
6:     end if
7: end for
```

E-eco performs state changes only when the time remaining on idle hosts are greater than the cost of transitions. Otherwise, no state change takes place. This operation can be seen in Algorithm 6.4, and it prevents hosts in RS change to IS or TS, and after having to return to a ready state to meet a new demand from RS.

6.4 Deployment for energy-efficient cloud orchestration

E-eco was developed considering a holistic view on a cloud scenario where such orchestrator has access to infrastructure resources. E-eco is deployed on the controller host and in the compute hosts. E-eco components running on the controller host are integrated into Openstack, and they are designed to manage the changing hosts for the sites discussed above, and check for centralized storage for virtual machine images. Components of e-eco running within compute hosts are implemented as SNMP agents, enabling or not the DVFS support. Besides, all hosts must have support for the ACPI sleep states, allowing them to be managed over a network. This control via the network is performed by e-eco from the controller host through the wakeup-on-lan tool.

The shaded area in Figure 6.3 shows the components involved in the choice of available hosts to support VM allocation when required. The component that performs dynamic allocation of VMs on the necessary resources is called Heat. It manages the list of available hosts through OpenStack's Nova service, which maintains a service server in the cloud controller host, and customer service in each cloud host to communicate with the controller host, updating their status.

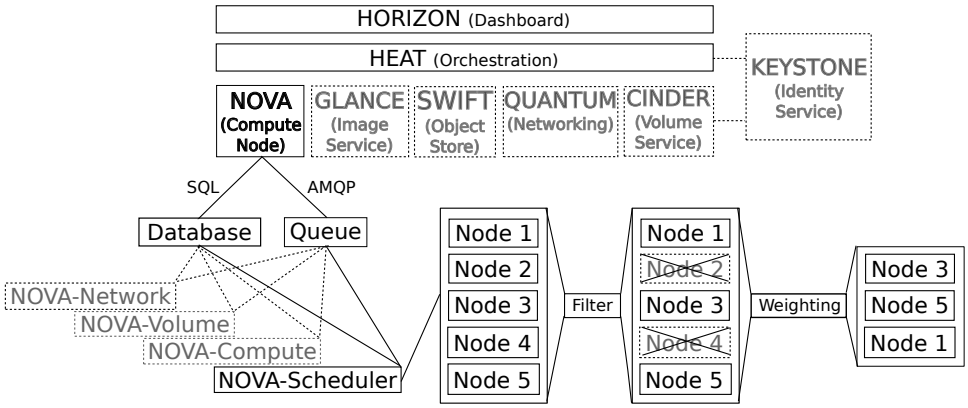


Figure 6.3 – Openstack scheduling architecture. The horizon panel shows the user interface for managing OpenStack services for both users and administrators. Nova provides a scalable calculation platform supporting the provision and management of large numbers of servers and VMs. This is accomplished by provisioning scheduling algorithms, weighing and filter the hosts available according to pre-determined metrics.

Because messages exchanged between hosts of the cloud and the controller host are performed via RPC (Remote Procedure Call), they are managed by an AMQP (Advanced Message Queuing Protocol) queue. Therefore, this component consists of a broker that stores information, in addition to acting on the hosts clients in order to manipulate the behavior of the VMs on them.

When there is a need to deploy new VMs on new hosts, the scheduler checks a list that contains the most compatible hosts to support a new instance. OpenStack supports various scheduling algorithms. Therefore each takes a different policy to the process of deciding which is the best available hosts, and Filter Scheduler is the most common among them. In this scheduler, the decision on which set of hosts is available and with enough setting to support demand is carried out in two stages: filtering and weighting. The filter presents the pool of available hosts, and the weight prioritizes these, based on policies such as the number of processors, memory, disk, network, etc.

OpenStack offers all necessary information for the e-eco power and performance management through Ceilometer APIs. This relationship of e-eco with the information provided by OpenStack can be seen in Figure 6.4. E-eco searches the necessary information to implement decisions aimed at energy saving, such as the number of hosts in several states, virtualization layer information, and internal information of each host.

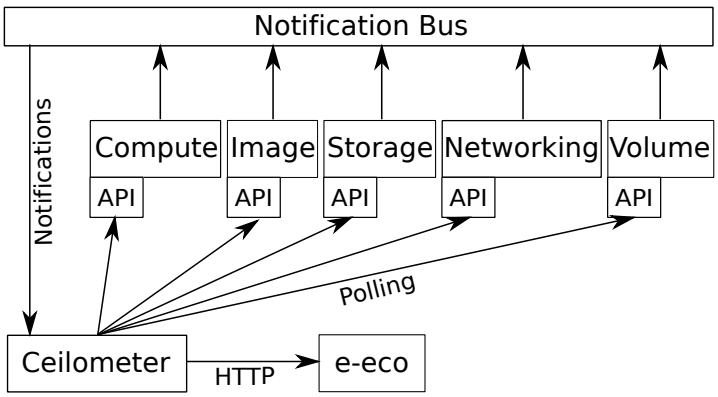


Figure 6.4 – The required information for e-eco to perform the changes in the algorithms presented below, such as the number of hosts running, the total number of available hosts, control flags on migration and DVFS, are captured by Ceilometer-API, the OpenStack module that gathers information from the various infrastructure services and components, such as the number of host on each state. The communication between e-eco and Ceilometer is performed via HTTP.

In more details, Figure 6.5 shows a sequence diagram of OpenStack components, when there is the allocation of a new physical host for a new virtual machine. As depicted in Figure 6.5, e-eco acts directly on the OpenStack component that defines which physical hosts is available for a new VM allocation. It does this by changing the perception of the queue of available hosts, pointing only to hosts on intermediate site (step 3 and 4). To do so, e-eco changes the queue of filter scheduler about which hosts are available for a new allocation. Before the Nova-scheduler component starts the instantiation of a VM on a new host, the host should be considered as available for an AMQP queue from Message Queue.

Figure 6.6(a) shows how the choice of the host that will receive a new virtual machine is performed. At this time, there is no distinction between the state in which the hosts are available. In this example, we use a power-agnostic environment, and the available hosts would be in idle state.

E-eco changes the queue of available hosts that are visible to the filter scheduler. This new queue is only offered with hosts that are in standby state under Intermediate Site. E-eco operation can be seen in Figure 6.6(b) where the filter scheduler still operates on a set of hosts, but only on the set of hosts offered as available by the e-eco. In this way, the filter scheduler can never see hosts turned off as available, and thus they always look for new hosts among those on standby.

Besides, e-eco presents the ability to decide which is the most energy efficient option without losing too much performance, either enable server consolidation or use DVFS in running hosts. This decision is based on the state shown by Swift, an OpenStack component that stores large-scale objects among the cloud hosts and can be configured so as centralized distributed. As we have discussed previously, if the storage is centralized, server consolidation is chosen and enabled through an internal flag in the core of Open-

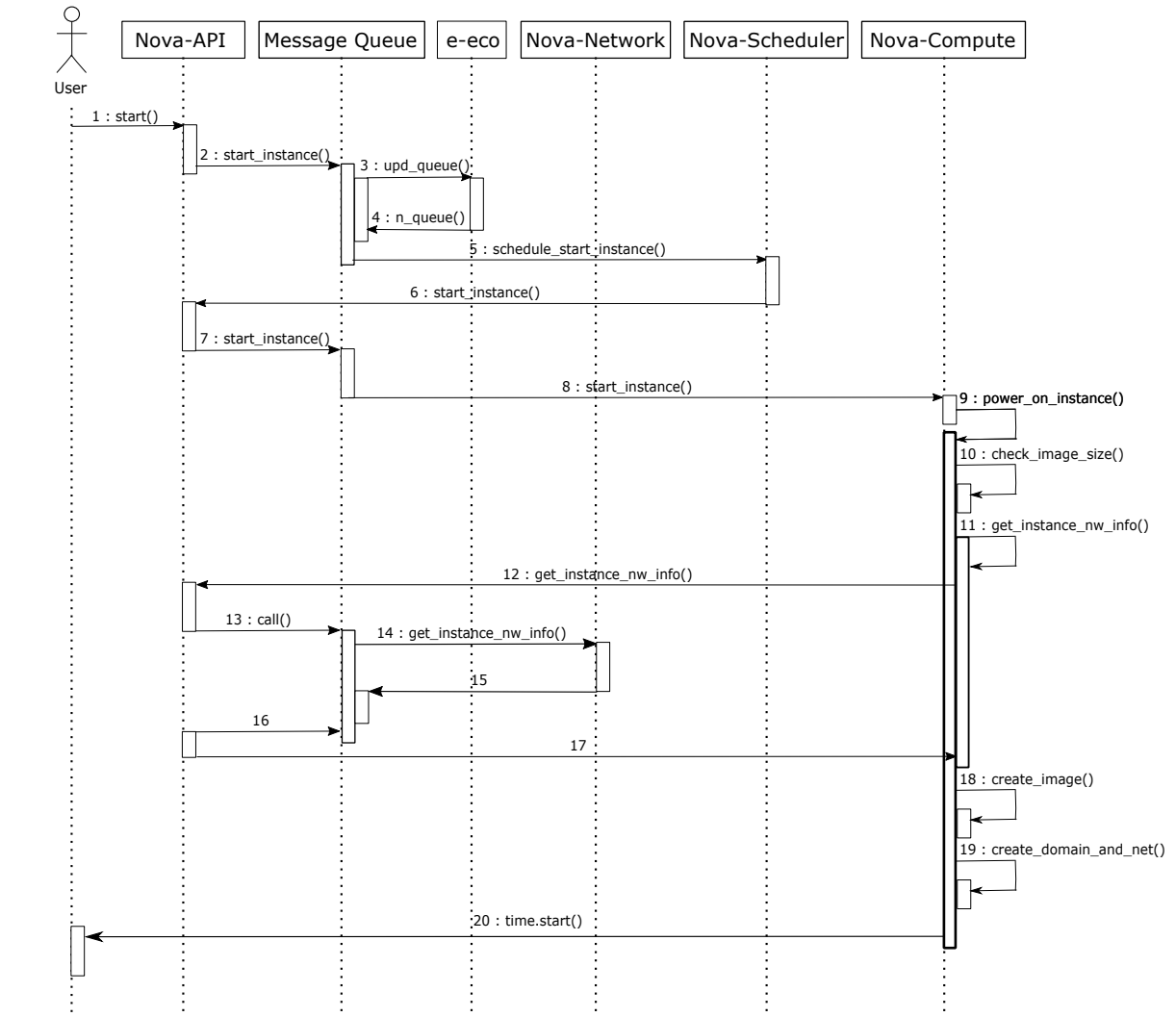
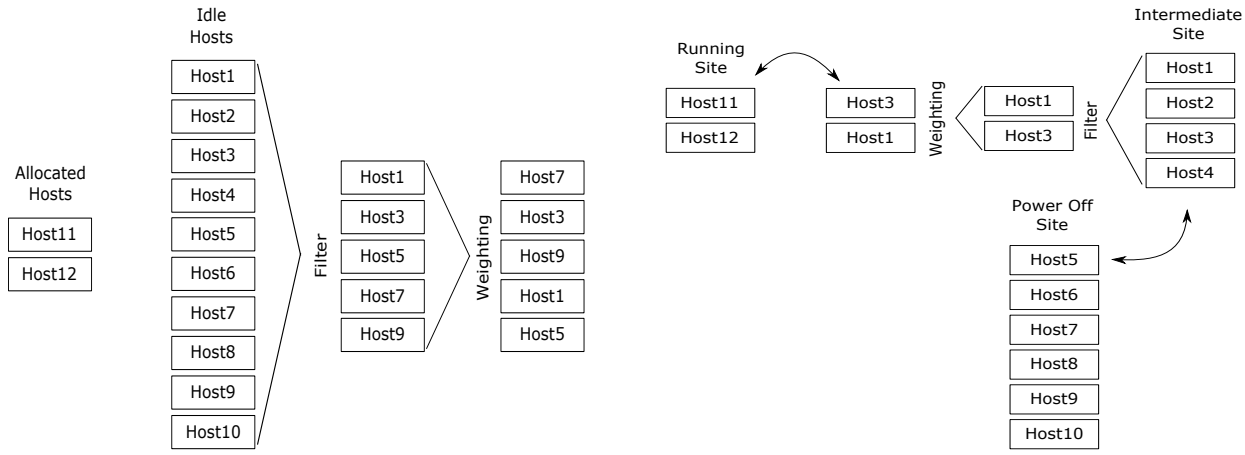


Figure 6.5 – Sequence diagram showing the instantiation process of a new virtual machine on a new physical host using e-eco.



(a) Openstack Filter Scheduler operation

(b) E-eco changing Openstack Filter Scheduler view

Figure 6.6 – Process of choosing a host where a new virtual machine will be instantiated in a traditional vision, and using e-eco.

Stack. When the storage is distributed, configuration commands are sent from the controller host to compute hosts via SNMP enabling DVFS.

6.5 Strategies evaluation and discussion

This section presents the evaluation of the improved trade-off between performance and energy savings enabled by e-eco in a real and a simulated cloud environment. Tests were performed in different conditions, as follows:

- **Power-Agnostic:** this case consists in an environment where there is no concern about energy saving, and therefore hosts are kept in one of two states: busy or idle. In this scenario, there is no VM consolidation nor reduction of processors frequency. Figure 6.7 illustrates such behavior.

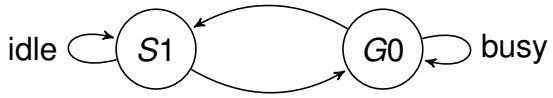


Figure 6.7 – Power-Agnostic Strategy's policies employed on one host.

- **Alvarruiz et al. [AdACH12]:** this case consists of the environment where hosts are kept in one of two states: running or off. In this scenario, VM consolidation and reduction of processors' frequency is also applied to the hosts. Figure 6.8 represents such behavior.

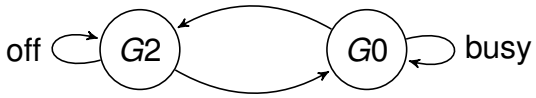


Figure 6.8 – Alvarruiz et al. Energy-saving Strategy's policies employed on one host.

- **Timeout Strategy [AIS08] [MGW09] [PB10]:** when the host in the G0 state becomes idle, it enters into the S3 state; the host returns immediately to the G0 state if it is requested; the host enters successively to a lower-power state if the timeout expires (300 seconds [IA15][RGKP10][LBT09]). Figure 6.9 represents such behavior.

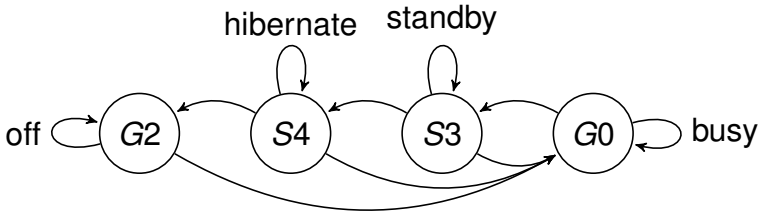


Figure 6.9 – Timeout Strategy's policies employed on one host.

- **e-eco:** this scenario consists of the management of hosts among execution, standby, and off states, to save energy, and at the same time impose less impact on application performance. Also, it applies VM consolidation or DVFS.

To carry out e-eco evaluations on a real cloud environment, the choice of a workload compatible with cloud environments is a determining factor. However, cloud traces are not available. The application behavior used in the test was based on Pucher et al. trace [PGWK15].

Besides, we follow the methodology adopted in the literature of using an online transaction processing (OLTP) database, widely used for tests in cloud environments, as it can be seen in [SCLG15], [LZO+13], and [JSS+15]. In this way, Sysbench [Kop14] was used as a benchmark for conducting evaluations of this work. This tool consists of a benchmark that performs multithreaded transactional queries on MySQL instances. In our assessments, we used the complex transactional mode with 75% of read operations and 25% of write operations. As a result, Sysbench provides performance information on the number of TPS achieved by the system. To mimic the behavior of scalable cloud applications, we used two test sets (customers and services) simultaneously. In total, customers perform 2000 sessions/connections on a database with 4.5GB, with an SLO set at 900 TPS. When the amount of TPS was less than agreed, the cloud environment should scale-out, launching new instances of the database on new hosts to support the growing demand. When the amount of TPS is greater than the established, the environment must scale-in to save resources.

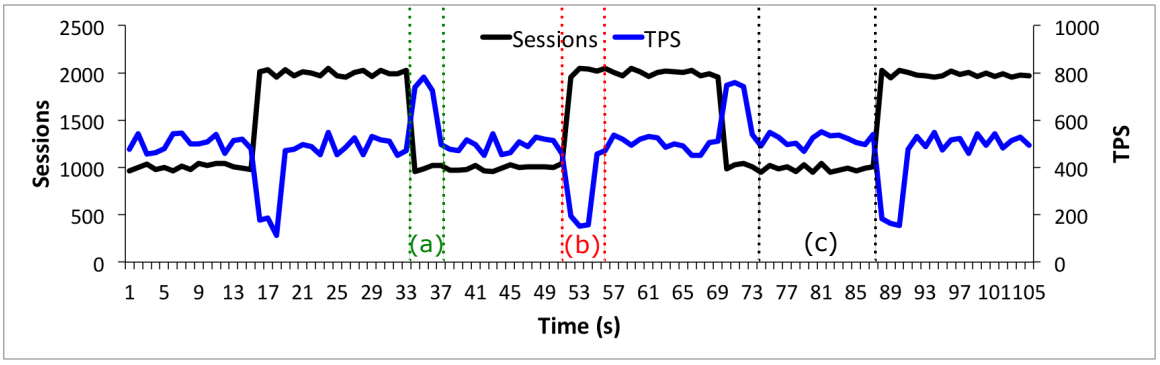


Figure 6.10 – Workload behavior phases to maintain a regular operation (c). In the case of underutilization, there is a time of SLA Saturation (a) until the environment can be adjusted to an amount sufficient of resources to meet demand. When there is a need for more resources due to an increase of requests, the performance drop causes SLA violation (b) until more resources are added to met the demand

Figure 6.10 shows the workload behavior, and defines three main phases: Regular Operation, SLA Saturation, and SLA Violation. Regular Operation is the maintenance of SLO set as 500 TPS. When there is under-utilization of resources (e.g., by reduction of customer requests), the extra resources make the TPS exceeds the SLA required by the

application. This case is called SLA Saturation, and the environment must be adjusted to the correct amount of resources so that again there is a Regular Operation, avoiding waste of resources. On the other hand, when the amount of TPS is reduced due to increased customer requests, more resources must be added to the environment to satisfy this new demand and return the environment to a Regular Operation. We call this time interval SLA Violation.

6.5.1 Experiments on a real cloud environment

E-eco system was deployed on a private cloud with 10 Citrix XenServer 6.2 hosts (Intel Xeon processors E5-4650 v2 2.40 GHz, 40/80 cores, 25 MB L3 Cache), using a Gigabit Ethernet network among them, and a centralized PowerVault MD3200i SAN Storage Array as a VM images storage, which is managed by OpenStack. OpenStack performs scale-out operations based on low-level metrics such as the use of the processor, memory, network, or a combination of these. As the analysis of the trade-off between performance and energy saving strategies must be carried out based on high-level metrics, we performed measurements where it was possible to estimate if a satisfactory use of resources is achieved with different levels of resource usage.

Table 6.1 shows that the Alvarruiz et al. strategy saves more energy, but it has a greater impact on application performance. The energy savings shown is due to all the idle hosts are necessarily off. The performance impact occurs for the same reason, because while hosts are restarting, the SLA remains in violation. The SLA saturation does not affect the performance directly, but the shutdown time impacts the use of resources and consequently impacts the energy consumption. EDP (Energy Delay Product) metric [BMS13] considers the latency together with the energy consumption through

$$EDP = Energy(Joules) x Delay(Seconds) \tag{6.6}$$

Such metric was used and it can clarify, summarize, and corroborate our findings correlating performance with energy consumption (the lower the EDP, the better the correlation between performance and energy saving).

The strategy presenting the smallest impact on application performance is the Power-Agnostic. This is because hosts are always on and ready to support new demands immediately. However, as the hosts remain connected, the power consumption is the highest among all strategies. The impact of these strategies on the performance are shown in Table 6.1, where we describe the total time of the test run, and how much time was spent on Regular Operation, SLA Violation, and SLA Saturation. It shows that Alvarruiz et al. reduces the performance of applications because they incur SLA violations for a longer time

Table 6.1 – Evaluation of e-eco conducted in a real cloud environment on 10 Xen hosts managed by Openstack, compared to other strategies.

Strategies	Execution Time	Energy Consumption	Cloud Usage			EDP (.10^8)
			Regular Operation	SLA Violation	SLA Saturation	
Power-Agnostic	5502s	646 Wh	95%	3%	2%	195
Alvarruiz et al.	5917s	455 Wh	50%	24%	26%	159
Timeout	5715s	531 Wh	70%	14%	16%	173
e-eco	5590s	485 Wh	92%	3%	5%	151

and have a long adjustment time of the environment during SLA saturation, which adds to the overhead on regular operation of the application.

The main goal of e-eco is to improve the trade-off between performance and energy savings. Results showed that for cloud environments, e-eco could save more energy than Power-Agnostic strategy, and have less impact on application performance than Alvarruiz et al. [AdACH12] strategy. This is because e-eco maintains a set of intermediate hosts that respond quickly to new demands, reducing the impact on performance, at the same time it saves energy while keeping the rest of hosts turned off. When compared to the timeout strategy, e-eco shows better performance and greater energy savings in addition to fewer SLA violations, thus enabling an increasingly regular operating time on cloud resources. The results of EDP metric can be seen in Table 6.1, where e-eco obtained the best results when compared to the other strategies.

The scale of cloud environments owned by typical cloud service provider is not easily reproduced in academic environments [BVWS14]. For example, it is estimated that Amazon EC2 holds more than 450,000 servers [Net15]. Although the tests have been carried out in a small-scale cloud (10 hosts), we believe that the performance gains and power savings provided by the e-eco remain for private cloud environments with a greater number of hosts. The works by Cameron [CGF07] and Zomaya [ZL12] make reference to the growth of energy consumption to the scale of hosts in a near-linear or linear rate. Performance may also be supported by the scalability of the cloud environment [CMK11]. This means that the energy savings and performance obtained by e-eco are expected to be proportional, for each cloud usage rate, to a higher number of hosts. Due to the absence of a significant amount of computing resources in our real environment, coupled with the reliability of the results presented in simulations, we decided to expand the size of our cloud to check the e-eco behavior in larger environments using Cloudsim [CRB+11].

6.5.2 Experiments on a simulated cloud environment

To expand the tests performed in a real environment, we adopted the CloudSim Simulator [CRB+11]. For the simulated tests, we used a trace based on Pucher et al. [PGWK15] using 1000 hosts. The tested strategies were the same evaluated in the previous section, with the addition of a virtual machine migration heuristic [BAB12] known as Minimum Migration Time (MMT) in cases when VM consolidation is needed. Such heuristic migrates a VM that wants the minimum time to complete a migration comparatively to other VMs with the possibility of being migrated to the host.

The first experiment concern the simulation of the real test of 10 hosts, from the previous experiments, to calibrate the CloudSim simulator. Results can be seen in the Table 6.2. The latency—the time spent during transitions—is compatible with the SLA Violation and Saturation (simulation results do not differ SLA violation and saturation, accounting a total) values previously presented in Table 6.1. The results showed a smaller standard deviation of up to 2%. Such results show that the simulator can achieve the same results shown in tests in the real environment, achieving a significant accuracy.

Table 6.2 – CloudSim simulation using 10 hosts.

Strategies	Execution Time	Energy Consumption	SLA Violation and Saturation	EDP (.10^8)
Power-Agnostic	5505s	673 Wh	4%	203
Alvarruiz et al.	5764s	477 Wh	41%	158
Timeout	5704s	537 Wh	28%	174
e-eco	5714s	381 Wh	4%	124

Table 6.3 shows the simulation results for a cloud environment with 1000 hosts using the Pucher et al. trace [PGWK15]. Results show that even in larger cloud environments, e-eco can maintain the best relationship between performance and power savings when compared to other strategies.

Table 6.3 – Simulation results among evaluated strategies on a cloud with 1000 hosts, taking into account the execution time in seconds, the energy consumed in watts, and EDP.

Strategies	Execution Time	Energy Consumption	SLA Violation and Saturation	EDP (.10^{11})
Power-Agnostic	5715s	57248 Wh	6%	187
Alvarruiz et al.	5864s	50720 Wh	39%	174
Timeout	5785s	53609 Wh	30%	179
e-eco	5720s	39114 Wh	5%	127

E-eco can achieve these results due to the fast response of hosts in IS when there is a new demand from RS. This occurs because hosts in IS respond faster than hosts in TS, such as the proposed by Alvarruiz et al. [AdACH12]. Furthermore, hosts in IS are in

a less deep state of energy-saving, yet consuming less energy than idle hosts on a Power-Agnostic strategy. The balance between performance and energy-saving by e-eco strategies is translated by EDP results in Table 6.3, where again, for larger data center environments, e-eco can maintain a better relationship between the two metrics than the Timeout strategy.

Table 6.4 – e-eco evaluations in an environment without server consolidation, where e-eco applied DVFS on processors.

Size	Testbed		Execution Time	Energy Consumption	SLA Violation and Saturation	EDP ($.10^8$)
	Real	Simulated				
10 hosts	x		5608s	493 Wh	7%	155
10 hosts		x	5615s	499 Wh	6%	157
1000 hosts		x	5729s	50826 Wh	6%	166

The tests presented were applied in cloud environments that allow server consolidation via migration of virtual machine (using a centralized storage to store virtual machine images). However, several large-scale cloud environments do not perform server consolidation. Table 6.4 shows the results of e-eco on such environments, using the same amount of hosts, traces, and strategies of the previous tests. Although this option offers increased EDP than the previously one, it still presents a better EDP than other strategies.

6.6 Concluding Remarks

The advantages brought by cloud computing have been promoting the establishment of data centers that support many different applications. Among the benefits of cloud computing, intelligent use of resources is a key factor, as through server virtualization, services can be scaled as they need. This management of resources impacts on operating costs for the provider of services, and among them, one of the most significant costs is the one with power consumption. Besides the consolidation of virtual machines intrinsically enabled by virtualized environments, several energy-saving techniques and mechanisms are used in cloud environments.

These solutions have different levels of intrusiveness in the cloud environment and offer many energy-saving levels. The problem with its use is the overhead they impose on others equally important metrics in an enterprise environment. One of the most important affected metrics, which directly involves customers' experience of quality, is the performance of applications.

The trade-off between performance and energy saving was studied and discussed in previous chapters of this work, and presented in many different levels of the data center, from individual hosts to sets of virtualized hosts connected by the network. Our tests showed that such trade-off is present when power-saving options are used and that intelligent management of these options impacts the performance penalty imposed by them. This intelligent

management becomes important when we consider that cloud services are offered through the Internet, and this can define the loyalty of a customer to the service.

In this direction, this chapter detailed the design of e-eco, an Energy-Efficient Cloud Orchestrator that improves the trade-off between energy savings and application performance through smart management of a set of power-saving options. A prototype has been implemented on real and simulated cloud environments, and tests have shown that e-eco maintains the balance between energy saving promoted with minimal impact on performance. Results of our evaluation demonstrated that e-eco can reduce energy consumption by up to 25% compared to power-agnostic approaches at a cost of only 6% of extra SLA violations. When compared to existing power-aware approaches, e-eco achieved the best relationship between performance and energy-saving, as expressed by EDP. These results showed that e-eco improves the trade-off between power savings and applications performance to enable a cloud environment that is at the same time economical and responsive.

Results achieved during the thesis showed that e-eco is feasible to improve the trade-off between energy savings and applications performance, from small to large-scale cloud environments. Such feasibility has been validated in real and simulated environments and showed that e-eco achieved better results than the other discussed and tested strategies.

7. CONCLUSION

Organizations collect data from a variety of sources, including commercial transactions, social networks, sensor information, and data transmitted from machine to machine. Data flows at an unprecedented speed and should be treated on time. Therefore are increasingly important computing environments that support the processing burden imposed by this new pattern. However, to perform the fast processing of this large amount of data, a significant amount of energy is required, either by the server operation or the cooling thereof. Currently, many solutions aimed at energy savings in data centers are being applied, but most of them impact on performance issues.

The main hypothesis suggested by this thesis is that a wise orchestration of energy saving techniques and mechanisms can save energy while reducing the impact on the performance of applications in cloud environments. This hypothesis was divided into several research questions that are correlated to the chapters of this dissertation. In this conclusion, we summarize the answers to these questions, we present findings and point out future research directions.

- *Research Question 1: Where are energy saving techniques and mechanisms available for cloud environments derived from and how is its management carried out?* We investigated technologies that allow energy savings in computing environments. Our study showed that there is a divergence about the terminology used to classify each of the energy-saving options provided both in hardware and in software. Therefore, we developed a terminology for the different components of an energy-saving solution for clouds to standardize the terms used in this dissertation. After this standardization of terms, we performed a terminology of related work to our research by using the proposed terminology, and to identify the real need for a solution that could address the proposed solution in our initial hypothesis. Besides, we studied the standard ACPI power management interface used by today's computers, which allowed us to understand the behavior of each sleep state from the operating system and the reduction in the frequency of the devices.

- *Research Question 2: What is the impact of energy-saving techniques and mechanisms on performance metrics in a single computer host?* We studied the behavior of sleep states, reduction of the processor frequency on a host, and the impact of such conditions on applications performance. Results showed that the deeper (regarding energy savings) the state, the greater is its latency to return to a full power state. An example of this trade-off can be based on the common view that turning off idle hosts can always save energy. If the environment presents a more organic behavior in which hosts are turned off, and on often, this option will consume more energy due to frequent context switches, as well as impact the performance of the application, adding

the overhead of transitions over this metric. The reduction in the processor frequency likewise reduces the number of instructions per second the processor can perform. In other words, this option can save energy, but increases the execution time of applications.

- *Research Question 3: What is the impact of energy-saving techniques and mechanisms on performance metrics on a set of computer hosts interconnected across the network?* Today's computing environments are not composed of only one host. In general, the computing environments are comprised of multiple parallel and distributed devices. Based on this premise, we evaluated every state analyzed above on an HPC cluster, and we proposed a solution to improve the trade-off between energy savings and application performance on a cluster. During the development of the Best-Fit Energy-Aware Algorithm, we realized that the times and power consumption during the transitions between some states have presented very similar values, which allows the choice of most appropriate state for each situation. Also, because our environment was an HPC cluster, reducing the processors frequency was not a viable solution as it reduced their processing capabilities. Therefore such algorithm chooses the best sleep state to replace the idle state for idle hosts, based on the job queue submitted to the cluster, reaching up to 55% energy savings with minimal impact on the jobs runtime. Therefore, the use of techniques and energy saving mechanisms, even in cluster environments where there is a high rate of use of resources, is a viable option when well managed.

- *Research Question 4: What is the difference between assessments previously achieved in this work compared to a cloud computing environment?* Best-Fit Energy-Aware Algorithm was developed to a cluster environment. However, this same algorithm could not be used directly to a cloud environment. An HPC cluster runs jobs using all available resources and HPC jobs present a finite completion time. Instead, the cloud provides access to a service, which runs steadily, and the burden on such service is related to the amount of requests carried out by customers. Therefore, we analyzed the impact of reduced processors frequency on the data center network, verifying that service requests are affected by such technique. Another important point is the migration of virtual machines, an option, a widely adopted practice in cloud environments. Our evaluations showed that there is an advantage to perform server consolidation when the cloud architecture provides a centralized storage capable of storing the VM images that will be instantiated on physical hosts. When there is no such images centralization, server consolidation is not recommended, and the best option is to reduce the frequency of processors.

- *Research Question 5: Is it possible to improve the trade-off between performance and energy saving in cloud data center environments?* Considering all the previous find-

ings, we proposed a solution (e-eco) that enhances the trade-off between energy savings and application performance in cloud environments through the previously studied techniques and mechanisms orchestration. Our solution is divided into two parts: the first part divides cloud in three sites with different states, and the number of hosts in each state is calculated based on a model in order to meet the SLA established; the second part decides, based on the data center architecture, which is the best option either carrying server consolidation or reduce the processor's frequency. Our evaluations have shown that e-eco improves the mentioned trade-off up to 25% compared with well-known strategies in the state-of-the-art. So yes, our results show that it is possible to improve such trade-off.

Based on the research work presented in this dissertation, our conclusion is that a correct orchestration of the techniques and mechanisms offered by cloud infrastructures allow a significant improvement of the trade-off between energy savings and application performance. This work studied the impact of energy savings techniques and mechanisms on the cloud application performance, by proposing an orchestration (e-eco) of them, so that application performance suffers the least impact due to the overhead of such techniques and mechanisms transitions. The results showed that e-eco achieves 25% energy savings and only 6% of SLA violations.

Additionally, we have listed some contributions:

1. a classification for the components of energy-saving solutions for data center environments.
2. study of the ACPI states behavior on a host and time and power consumption evaluation during their transitions;
3. proposal for a performance-aware energy-saving strategy for a cluster of computers.
4. impact analysis of the energy-saving states on the hosts network.
5. implementation of a performance-aware orchestration of energy-saving techniques and mechanisms in cloud environments.

7.1 Future Research Directions

There are several possible directions for future research based on this work. We suggest some possibilities for future research as follows.

7.1.1 Heuristics for VM placement

E-eco operates at a level above certain decisions of the infrastructure, such as the manner and place where virtual machines will be migrated. This dissertation used a heuristic known as Minimum Migration Time (MMT), one of the heuristics presented by Beloglazov et al. [BB12], which aims to optimize performance during server consolidation. Other heuristics that decide in what ways virtual machines must be migrated can be tested along with e-eco, and it can provide even better results of EDP.

7.1.2 Redundant links

We showed that in cloud environments without centralized storage for VM images, the most energy-efficient decision is not to perform VM consolidation. However, most of the today's data centers have redundant network paths that may support link aggregation mechanisms to increase the communication channel between hosts and enable faster traffic of such VM images. Thus, the investigation of how such mechanism may affect the decisions made by e-eco can be interesting.

7.1.3 Impact of new devices

Every day, more devices are being developed with the bias for energy efficiency. Additionally, some devices do not have more boot overheads when compared with traditional ones, such as the Solid State Disk (SSD). Based on this, new sleep states as Hibernate can be incorporated into the model, given such new disk have smaller delays for state savings and thus do not impose extra overhead for hibernation compared to the standby state.

7.1.4 Different frequencies on independent cores

Currently, only two processors (Intel Nehalem and Transmeta Crusoe TM5800) allow the change of the cores frequency in an independent manner. Both processors are not standard in most data centers. However, the adoption of this technology should change the decisions of the e-eco when it acts on RS, by setting different frequencies to different virtual machines on the same host. Another option would be the separation of RS on hosts different frequency sites, where virtual machines can migrate among them depending on their load.

7.1.5 Containers

Containers consist of virtualized environments that have near-native performance, increasingly present in large-scale environments. Although this new model still has limitations on the isolation capabilities, it can be a great opportunity for environments where performance is more important, because containers do not present instantiation time, such as traditional virtual machines.

REFERENCES

[ABB+11] Auweter, A.; Bode, A.; Brehm, M.; Huber, H.; Kranzlmüller, D. “Principles of energy efficiency in high performance computing”. In: Proceedings of the First International Conference on Information and Communication on Technology for the Fight Against Global Warming, 2011, pp. 18–25.

[AdACH12] Alvarruiz, F.; de Alfonso, C.; Caballer, M.; Hernandez, V. “An energy manager for high performance computer clusters”. In: Proceedings of the IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA), 2012, pp. 231 –238.

[AIS08] Augustine, J.; Irani, S.; Swamy, C. “Optimal power-down strategies”, *SIAM Journal of Computing*, vol. 37–5, Jan 2008, pp. 1499–1516.

[AN11] Agarwal, S.; Nath, A. “Green computing - a new horizon of energy efficiency and electronic waste minimization: A global perspective”. In: Proceedings of the International Conference on Communication Systems and Network Technologies, 2011, pp. 688–693.

[APT14] Alboaneen, D. A.; Pranggono, B.; Tianfield, H. “Energy-aware virtual machine consolidation for cloud data centers”. In: Proceedings of the 2014 7th IEEE/ACM International Conference on Utility and Cloud Computing, 2014, pp. 1010–1015.

[BAB12] Beloglazov, A.; Abawajy, J.; Buyya, R. “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing”, *Future Generation Computer Systems*, vol. 28–5, May 2012, pp. 755–768.

[BAK99] Ben-Ari, M.; Kolikant, Y. B.-D. “Thinking parallel: The process of learning concurrency”, *SIGCSE Bulletin*, vol. 31–3, Jun 1999, pp. 13–16.

[BB10] Beloglazov, A.; Buyya, R. “Energy efficient resource management in virtualized cloud data centers”. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010, pp. 826–831.

[BB12] Beloglazov, A.; Buyya, R. “Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers”, *Concurrency Computing : Practice and Experience*, vol. 24–13, Sep 2012, pp. 1397–1420.

[BCL+11] Byrne, J.; Chang, J.; Lim, K. T.; Ramirez, L.; Ranganathan, P. “Power-efficient networking for balanced system designs: Early experiences with pcie”. In: Proceedings of the 4th Workshop on Power-Aware Computing and Systems, 2011, pp. 3:1–3:5.

[BJL+06] Brecht, T.; Janakiraman, G. J.; Lynn, B.; Saletore, V.; Turner, Y. “Evaluating network processing efficiency with processor partitioning and asynchronous i/o”. In: Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems, 2006, pp. 265–278.

[BL12] Bays, W.; Lange, K.-D. “Spec: driving better benchmarks”. In: Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, 2012, pp. 249–250.

[BMS13] Blem, E.; Menon, J.; Sankaralingam, K. “Power struggles: Revisiting the RISC vs. CISC debate on contemporary arm and x86 architectures”. In: Proceedings of the IEEE 19th International Symposium on High Performance Computer Architecture (HPCA2013), 2013, pp. 1–12.

[Boh98] Bohr, M. “Silicon trends and limits for advanced microprocessors”, *Communications ACM*, vol. 41–3, Mar 1998, pp. 80–87.

[Bri15] Brito, N. “T50”. Capturado em: http://t50.sourceforge.net, Oct 2015.

[BSB+10] Bolte, M.; Sievers, M.; Birkenheuer, G.; Niehörster, O.; Brinkmann, A. “Non-intrusive virtualization management using libvirt”. In: Proceedings of the Conference on Design, Automation and Test in Europe, 2010, pp. 574–579.

[Buy99] Buyya, R. “High Performance Cluster Computing: Architectures and Systems”. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999.

[BVWS14] Barker, A.; Varghese, B.; Ward, J. S.; Sommerville, I. “Academic cloud computing research: Five pitfalls and five opportunities”. In: Proceedings of the 6th USENIX Conference on Hot Topics in Cloud Computing, 2014, pp. 2–2.

[Cam14] Cameron, K. W. “Energy efficiency in the wild: Why datacenters fear power management”, *Computer*, vol. 47–11, Nov 2014, pp. 89–92.

[CAT+01] Chase, J. S.; Anderson, D. C.; Thakar, P. N.; Vahdat, A. M.; Doyle, R. P. “Managing energy and server resources in hosting centers”. In: Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles, 2001, pp. 103–116.

[CFF14] Corradi, A.; Fanelli, M.; Foschini, L. “VM consolidation”, *Future Generation Computing Systems*, vol. 32–C, Mar 2014, pp. 118–127.

[CFH+05] Clark, C.; Fraser, K.; Hand, S.; Hansen, J. G.; Jul, E.; Limpach, C.; Pratt, I.; Warfield, A. “Live migration of virtual machines”. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2, 2005, pp. 273–286.

[CGF07] Cameron, K. W.; Ge, R.; Feng, X. “Designing computational clusters for performance and power”. In: *Architectural Issues*, Zelkowitz, M. V. (Editor), Elsevier, 2007, *Advances in Computers*, vol. 69, pp. 89 – 153.

[CGV+11] Chen, Q.; Grosso, P.; Veldt, K. v. d.; Laat, C. d.; Hofman, R.; Bal, H. “Profiling energy consumption of VMs for green cloud computing”. In: Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, 2011, pp. 768–775.

[CLQ08] Casanova, H.; Legrand, A.; Quinson, M. “Simgrid: A generic framework for large-scale distributed experiments”. In: Proceedings of the Tenth International Conference on Computer Modeling and Simulation, 2008, pp. 126–131.

[CMK11] Chieu, T.; Mohindra, A.; Karve, A. “Scalability and performance of web applications in a compute cloud”. In: Proceedings of the 2011 IEEE 8th International Conference on e-Business Engineering (ICEBE), 2011, pp. 317–323.

[CRB+11] Calheiros, R. N.; Ranjan, R.; Beloglazov, A.; De Rose, C. A. F.; Buyya, R. “Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms”, *Software Practice and Experience*, vol. 41–1, Jan 2011, pp. 23–50.

[DK13a] Delgado, B.; Karavanic, K. “Performance implications of system management mode”. In: Proceedings of the 2013 IEEE International Symposium on Workload Characterization (IISWC), 2013, pp. 163–173.

[DK13b] Delimitrou, C.; Kozyrakis, C. “The netflix challenge: Datacenter edition”, *IEEE Computing Architecture Letters*, vol. 12–1, Jan 2013, pp. 29–32.

[DL11] Dongarra, J.; Luszczek, P. “Linpack benchmark”. In: *Encyclopedia of Parallel Computing*, Springer, 2011, pp. 1033–1036.

[DQLW15] Ding, Y.; Qin, X.; Liu, L.; Wang, T. “Energy efficient scheduling of virtual machines in cloud with deadline constraint”, *Future Generation Computer Systems*, vol. 50–C, Sep 2015, pp. 62–74.

[DSI10] Duy, T. V. T.; Sato, Y.; Inoguchi, Y. “Performance evaluation of a green scheduling algorithm for energy savings in cloud computing”. In: Proceedings of the IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010, pp. 1–8.

[DZJW15] Dong, Y.; Zhou, L.; Jin, Y.; Wen, Y. “Improving energy efficiency for mobile media cloud via virtual machine consolidation”, *Mobile Network Applications*, vol. 20–3, Jun 2015, pp. 370–379.

[EE11] Eyerman, S.; Eeckhout, L. “Fine-grained DVFS using on-chip regulators”, *ACM Transactions on Architecture and Code Optimization*, vol. 8–1, Feb 2011, pp. 1:1–1:24.

[ESSA+12] El-Sayed, N.; Stefanovici, I. A.; Amvrosiadis, G.; Hwang, A. A.; Schroeder, B. “Temperature management in data centers: why some (might) like it hot”. In: Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems, 2012, pp. 163–174.

[FFCdM11] Fischer, A.; Fessi, A.; Carle, G.; de Meer, H. “Wide-area virtual machine migration as resilience mechanism”. In: Proceedings of the 2011 30th IEEE Symposium on Reliable Distributed Systems Workshops (SRDSW), 2011, pp. 72–77.

[FGJN+10] Fitz Gibbon, A.; Joiner, D. A.; Neeman, H.; Peck, C.; Thompson, S. “Teaching high performance computing to undergraduate faculty and undergraduate students”. In: Proceedings of the 2010 TeraGrid Conference, 2010, pp. 7:1–7:7.

[Fou15] Foundation, A. “Apache-bench”. Capturado em: http://httpd.apache.org/docs/2.4/programs/ab.html, Oct 2015.

[FRM+10] Feller, E.; Rilling, L.; Morin, C.; Lottiaux, R.; Leprince, D. “Snooze: A scalable, fault-tolerant and distributed consolidation manager for large-scale clusters”. In: Proceedings of the 2010 IEEE/ACM International Conference on Green Computing and Communications & International Conference on Cyber, Physical and Social Computing, 2010, pp. 125–132.

[GAL14] Gebrehiwot, M. E.; Aalto, S. A.; Lassila, P. “Optimal sleep-state control of energy-aware m/g/1 queues”. In: Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools, 2014, pp. 82–89.

[GB12] Galante, G.; Bona, L. C. E. d. “A survey on cloud computing elasticity”. In: Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing, 2012, pp. 263–270.

[GMAMPMJGI06] Gil-Martinez-Abarca, J.; Macia-Perez, F.; Marcos-Jorquera, D.; Gilart-Iglesias, V. “Wake on lan over internet as web service”. In: Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation, 2006. ETFA ’06, 2006, pp. 1261–1268.

[GMC+13] Guerout, T.; Monteil, T.; Costa, G. D.; Calheiros, R. N.; Buyya, R.; Alexandru, M. “Energy-aware simulation with DVFS.”, *Simulation Modelling Practice and Theory*, vol. 39, 2013, pp. 76–91.

[Gro03] Grover, A. “Modern system power management”, *Queue*, vol. 1–7, Oct 2003, pp. 66–72.

[GWGP13] Gao, Y.; Wang, Y.; Gupta, S. K.; Pedram, M. “An energy and deadline aware resource provisioning, scheduling and optimization framework for cloud systems”. In: Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, 2013, pp. 31:1–31:10.

[HDC+05] Heath, T.; Diniz, B.; Carrera, E. V.; Meira, Jr., W.; Bianchini, R. “Energy conservation in heterogeneous server clusters”. In: Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming, 2005, pp. 186–195.

[HGGG12] Han, R.; Guo, L.; Ghanem, M. M.; Guo, Y. “Lightweight resource scaling for cloud applications”. In: Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012), 2012, pp. 644–651.

[HKG+07] Hanson, H.; Keckler, S. W.; Ghiasi, S.; Rajamani, K.; Rawson, F.; Rubio, J. “Thermal response to DVFS: analysis with an intel pentium m”. In: Proceedings of the 2007 international symposium on Low power electronics and design, 2007, pp. 219–224.

[HPD+14] Huergo, R. S.; Pires, P. F.; Delicato, F. C.; Costa, B.; Cavalcante, E.; Batista, T. “A systematic survey of service identification methods”, *Service Oriented Computing and Applications*, vol. 8–3, Sep 2014, pp. 199–219.

[HR10] Humphries, C.; Ruth, P. “Towards power efficient consolidation and distribution of virtual machines”. In: Proceedings of the 48th Annual Southeast Regional Conference, 2010, pp. 75:1–75:6.

[HS08] Horvath, T.; Skadron, K. “Multi-mode energy management for multi-tier server clusters”. In: Proceedings of the 17th international conference on Parallel architectures and compilation techniques, 2008, pp. 270–279.

[IA15] Intel; Agency, U. E. P. “Energy star* version 5.0 system implementation whitepaper”. Accessed on October 2015., 2015.

[ILA+11] Isci, C.; Liu, J.; Abali, B.; Kephart, J. O.; Kouloheris, J. “Improving server utilization using fast virtual machine migration”, *IBM Journal of Research and Development*, vol. 55–6, Nov 2011, pp. 365–376.

[IMK+13] Isci, C.; McIntosh, S.; Kephart, J.; Das, R.; Hanson, J.; Piper, S.; Wolford, R.; Brey, T.; Kantner, R.; Ng, A.; Norris, J.; Traore, A.; Frissora, M. “Agile, efficient virtualization power management with low-latency server power states”, *ACM SIGARCH Computer Architecture News*, vol. 41–3, Jun 2013, pp. 96–107.

[JS10] Junior, A.; Sofia, R. “Energy-efficient routing in user-centric environments”. In: Proceedings of the 2010 IEEE/ACM Int’L Conference on Green Computing and Communications & Int’L Conference on Cyber, Physical and Social Computing, 2010, pp. 436–441.

[JSS+15] Jia, Q.; Shen, Z.; Song, W.; van Renesse, R.; Weatherspoon, H. “Supercloud: Opportunities and challenges”, *SIGOPS Operating System Review*, vol. 49–1, Jan 2015, pp. 137–141.

[KG14] Khomonenko, A. D.; Gindin, S. I. “Stochastic models for cloud computing performance evaluation”. In: Proceedings of the 10th Central and Eastern European Software Engineering Conference in Russia, 2014, pp. 20:1–20:6.

[KJF11] Kipp, A.; Jiang, T.; Fugini, M. “Green metrics for energy-aware it systems”. In: Proceedings of the 2011 International Conference on Complex, Intelligent, and Software Intensive Systems, 2011, pp. 241–248.

[KL14] Kong, F.; Liu, X. “A survey on green-energy-aware power management for datacenters”, *ACM Computing Surveys*, vol. 47–2, Nov 2014, pp. 30:1–30:38.

[KM08] Kaxiras, S.; Martonosi, M. “Computer Architecture Techniques for Power-Efficiency”. Morgan and Claypool Publishers, 2008, 1st ed..

[KMA+10] Krioukov, A.; Mohan, P.; Alspaugh, S.; Keys, L.; Culler, D.; Katz, R. H. “Napsac: Design and implementation of a power-proportional web cluster”. In: Proceedings of the First ACM SIGCOMM Workshop on Green Networking, 2010, pp. 15–22.

[Kop14] Kopytov, A., “Sysbench: A system performance benchmark”, dez 2014, https://launchpad.net/sysbench.

[KZS11] Kolpe, T.; Zhai, A.; Sapatnekar, S. S. “Enabling improved power management in multicore processors through clustered DVFS”. In: Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011, 2011, pp. 1–6.

[LBT09] Lammie, M.; Brenner, P.; Thain, D. “Scheduling grid workloads on multicore clusters to minimize energy and maximize performance”. In: Proceedings of the 10th IEEE/ACM International Conference on Grid Computing, 2009, 2009, pp. 145–152.

[LF03] Lublin, U.; Feitelson, D. G. “The workload on parallel supercomputers: modeling the characteristics of rigid jobs”, *Journal of Parallel and Distributed Computing*, vol. 63–11, Nov 2003, pp. 1105–1122.

[LGW05] Li, H.; Groep, D.; Wolters, L. “Workload characteristics of a multi-cluster supercomputer”. In: Proceedings of the 10th international conference on Job Scheduling Strategies for Parallel Processing, 2005, pp. 176–193.

[LKLK09] Lee, W. Y.; Ko, Y. W.; Lee, H.; Kim, H. “Energy-efficient scheduling of a real-time task on DVFS-enabled multi-cores”. In: Proceedings of the 2009 International Conference on Hybrid Information Technology, 2009, pp. 273–277.

[LO10] Lefèvre, L.; Orgerie, A.-C. “Designing and evaluating an energy efficient cloud”, *Journal of Supercomputing*, vol. 51–3, Mar 2010, pp. 352–373.

[LTM+12] Liu, F.; Tong, J.; Mao, J.; Bohn, R.; Messina, J.; Badger, L.; Leaf, D. “NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology (Special Publication 500-292)”. USA: CreateSpace Independent Publishing Platform, 2012.

[LW14] Li, D.; Wu, J. “Energy-aware scheduling for aperiodic tasks on multi-core processors”. In: Proceedings of the 43rd International Conference on Parallel Processing (ICPP), 2014, 2014, pp. 361–370.

[LZ10] Liu, Y.; Zhu, H. “A survey of the research on power management techniques for high-performance systems”, *Software: Practice and Experience*, vol. 40–11, Oct 2010, pp. 943–964.

[LZO+13] Li, Z.; Zhang, H.; O’Brien, L.; Cai, R.; Flint, S. “On evaluating commercial cloud services”, *Journal of Systems and Software*, vol. 86–9, Sep 2013, pp. 2371–2393.

[Mal11] Malewicz, G. “Beyond mapreduce”. In: Proceedings of the Second International Workshop on MapReduce and Its Applications, 2011, pp. 25–26.

[Mat07] Mateescu, G. “Overcoming the processor communication overhead in mpi applications”. In: Proceedings of the 2007 spring simulation multiconference - Volume 2, 2007, pp. 375–378.

[MD15] Maccio, V.; Down, D. “On optimal policies for energy-aware servers”, *Performance Evaluation*, vol. 90–C, Aug 2015, pp. 36–52.

[Med05] Medernach, E. “Workload analysis of a cluster in a grid environment”. In: Proceedings of the 11th international conference on Job Scheduling Strategies for Parallel Processing, 2005, pp. 36–61.

[MG11] Matthíasdóttir, A.; Geirsson, H. J. “The novice problem in computer science”. In: Proceedings of the 12th International Conference on Computer Systems and Technologies, 2011, pp. 570–576.

[MGW09] Meisner, D.; Gold, B. T.; Wenisch, T. F. “Powernap: Eliminating server idle power”. In: Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems, 2009, pp. 205–216.

[MKD04] McCallister, M.; Kutyna, M.; Delgado, D. “Implementing high-performance grid computing in student computer labs”. In: Proceedings of the 32Nd Annual ACM SIGUCCS Conference on User Services, 2004, pp. 360–360.

[MNDR10] Maillard, N.; Navaux, P.; De Rose, C. “Energy-aware scheduling of parallel programs”. In: Proceedings of the Conferencia Latino Americana de Computación de Alto Rendimiento, 2010, pp. 95–101.

[Mon12] Mone, G. “Redesigning the data center”, *Communications ACM*, vol. 55–10, Oct 2012, pp. 14–16.

[MVM09] Miller, F. P.; Vandome, A. F.; McBrewster, J. “Moore’s Law: History of Computing Hardware, Integrated Circuit, Accelerating Change, Amdahl’s Law, Metcalfe’s Law, Mark Kryder, Jakob Nielsen (Usability Consultant), Wirth’s Law”. Alpha Press, 2009.

[MWT+12] Min, A. W.; Wang, R.; Tsai, J.; Ergin, M. A.; Tai, T.-Y. C. “Improving energy efficiency for mobile platforms by exploiting low-power sleep states”. In: Proceedings of the 9th Conference on Computing Frontiers, 2012, pp. 133–142.

[NCS09] Niyato, D.; Chaisiri, S.; Sung, L. B. “Optimal power management for server farm to support green computing”. In: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009, pp. 84–91.

[Net15] Netcraft, “Amazon web services’ growth unrelenting”, aug 2015, http://news.netcraft.com/archives/2013/05/20/amazon-web-services-growth-unrelenting.html.

[NLA15] NLANR/DAST. “Iperf”. Capturado em: http://iperf.sourceforge.net, Oct 2015.

[NS07] Nathuji, R.; Schwan, K. “Virtualpower: coordinated power management in virtualized enterprise systems”, *SIGOPS Operating Systems Review*, vol. 41–6, Oct 2007, pp. 265–278.

[NWG+09] Nurmi, D.; Wolski, R.; Grzegorczyk, C.; Obertelli, G.; Soman, S.; Youseff, L.; Zagorodnov, D. “The eucalyptus open-source cloud-computing system”. In: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009, pp. 124–131.

[PB10] Ponciano, L.; Brasileiro, F. “On the impact of energy-saving strategies in opportunistic grids”. In: Proceedings of the 2010 11th IEEE/ACM International Conference on Grid Computing (GRID), 2010, pp. 282–289.

[Pep11] Pepple, K. “Deploying OpenStack”. " O’Reilly Media, Inc.", 2011.

[PGWK15] Pucher, A.; Gul, E.; Wolski, R.; Krintz, C. “Using trustworthy simulation to engineer cloud schedulers”. In: Proceedings of the IEEE International Conference on Cloud Engineering (IC2E), 2015, 2015, pp. 256–265.

[PJDS14] Pietri, I.; Juve, G.; Deelman, E.; Sakellariou, R. “A performance model to estimate execution time of scientific workflows on the cloud”. In: Proceedings of the 9th Workshop on Workflows in Support of Large-Scale Science, 2014, pp. 11–19.

[RB14] Rosado, T.; Bernardino, J. “An overview of openstack architecture”. In: Proceedings of the 18th International Database Engineering & Applications Symposium, 2014, pp. 366–367.

[RDR14a] Rossi, F., C. E. X. M. F. T.; De Rose, C. “Green software development for multicore architectures”. In: Proceedings of the IEEE Symposium on Computers and Communication (ISCC), 2014, pp. 01–06.

[RDR14b] Rossi, D., C. M. F. T.; De Rose, C. “Evaluating the trade-off between DVFS energy-savings and virtual networks performance”. In: Proceedings of the Thirteenth International Conference on Networks (ICN), 2014, pp. 274–279.

[RDR15a] Rossi, F.; De Oliveira, I. C. R. B. R.; De Rose, C. “Non-invasive estimation of cloud applications performance via hypervisor’s operating systems counters”. In: Proceedings of the Fourteenth International Conference on Networks (ICN), 2015, pp. 177–184.

[RDR15b] Rossi, F., S. M. D. O. I.; De Rose, C. “Modeling power consumption for DVFS policies”. In: Proceedings of the IEEE International Symposium on Circuits Systems (ISCAS), 2015, pp. 1879–1882.

[RDR15c] Rossi, F., X. M. m. Y.; De Rose, C. “On the impact of energy-efficient strategies in hpc clusters”. In: Proceedings of the 23rd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), 2015, pp. 17–21.

[RFAA13] Rajasekaran, S.; Fiondella, L.; Ahmed, M.; Ammar, R. A. “Multicore Computing: Algorithms, Architectures, and Applications”. Chapman & Hall/CRC, 2013, 1st ed..

[RGKP10] Reich, J.; Goraczko, M.; Kansal, A.; Padhye, J. “Sleepless in seattle no longer”. In: Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference, 2010, pp. 17–17.

[San15] Sanfilippo, S. “Hping”. Capturado em: http://www.hping.org, Oct 2015.

[SCLG15] Scheuner, J.; Cito, J.; Leitner, P.; Gall, H. “Cloud workbench: Benchmarking iaas providers based on infrastructure-as-code”. In: Proceedings of the 24th International Conference on World Wide Web Companion, 2015, pp. 239–242.

[SDM11] Shalf, J.; Dosanjh, S.; Morrison, J. “Exascale computing technology challenges”. In: Proceedings of the 9th International Conference on High Performance Computing for Computational Science, 2011, pp. 1–25.

[Sei98] Seifert, R. “Gigabit Ethernet: Technology and Applications for High-Speed LANs”. Addison-Wesley Longman Publishing Co., Inc., 1998.

[SFBS+12] Silva-Filho, A. G.; Bezerra, P. T. L.; Silva, F. Q. B.; Junior, A. L. O. C.; Santos, A. L. M.; Costa, P. H. R.; Miranda, R. C. G. “Energy-aware technology-based DVFS mechanism for the android operating system”. In: Proceedings of the 2012 Brazilian Symposium on Computing System Engineering, 2012, pp. 184–187.

[Sha02] Shanley, T. “Infiniband”. Addison-Wesley Longman Publishing Co., Inc., 2002.

[SKKE11] Spiliopoulos, V.; Keramidas, G.; Kaxiras, S.; Efstathiou, K. “Poster: DVFS management in real-processors”. In: Proceedings of the international conference on Supercomputing, 2011, pp. 373–373.

[SLD+15] Shen, D.; Luo, J.; Dong, F.; Fei, X.; Wang, W.; Jin, G.; Li, W. “Stochastic modeling of dynamic right-sizing for energy-efficiency in cloud data centers”, *Future Generation Computer Systems*, vol. 48–C, Jul 2015, pp. 82–95.

[SLM10] Santana, C.; Leite, J. C. B.; Mossé, D. “Load forecasting applied to soft real-time web clusters”. In: Proceedings of the 2010 ACM Symposium on Applied Computing, 2010, pp. 346–350.

[SML10] Sahoo, J.; Mohapatra, S.; Lath, R. “Virtualization: A survey on concepts, taxonomy and associated security issues”. In: Proceedings of the 2010 Second International Conference on Computer and Network Technology, 2010, pp. 222–226.

[SMS+04] Satran, J.; Meth, K.; Sapuntzakis, C.; Chadalapaka, M.; Zeidner, E. “Internet small computer systems interface”, Technical Report, 2004.

[Som01] Sommerville, I. “Engenharia de Software”. Person Education, 201, vol. 9.

[SS05] Sabin, G.; Sadayappan, P. “Unfairness metrics for space-sharing parallel job schedulers”. In: Proceedings of the 11th international conference on Job Scheduling Strategies for Parallel Processing, 2005, pp. 238–256.

[SS12] Sanders, P.; Speck, J. “Energy efficient frequency scaling and scheduling for malleable tasks”. In: Proceedings of the 18th international conference on Parallel Processing, 2012, pp. 167–178.

[SV13] Suleiman, B.; Venugopal, S. “Modeling performance of elasticity rules for cloud-based applications”. In: Proceedings of the 2013 17th IEEE

International Enterprise Distributed Object Computing Conference, 2013, pp. 201–206.

[TLCS12] Tiwari, A.; Laurenzano, M. A.; Carrington, L.; Snavely, A. “Auto-tuning for energy usage in scientific applications”. In: Proceedings of the 2011 international conference on Parallel Processing - Volume 2, 2012, pp. 178–187.

[UTCL12] Utrera, G.; Tabik, S.; Corbalan, J.; Labarta, J. “A job scheduling approach for multi-core clusters based on virtual malleability”. In: Proceedings of the 18th international conference on Parallel Processing, 2012, pp. 191–203.

[vdSvMC12] van der Spoel, D.; van Maaren, P. J.; Caleman, C. “Gromacs molecule & liquid database”, *Bioinformatics*, vol. 28–5, Mar 2012, pp. 752–753.

[Vog08] Vogels, W. “Beyond server consolidation”, *Queue*, vol. 6–1, Jan 2008, pp. 20–26.

[VS11] Villegas, D.; Sadjadi, S. M. “Mapping non-functional requirements to cloud applications.” In: Proceedings of the International Conference on Software Engineering and Knowledge Engineering, 2011, pp. 527–532.

[WCC14] Wu, C.-M.; Chang, R.-S.; Chan, H.-Y. “A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters”, *Future Generation Computer Systems*, vol. 37–0, 2014, pp. 141 – 147.

[Wei98] Weil, A. “Utilization and predictability in scheduling the ibm sp2 with backfilling”. In: Proceedings of the 12th. International Parallel Processing Symposium on International Parallel Processing Symposium, 1998, pp. 542–.

[WT10] Wakamatsu, H.; Takahashi, T. “Communication system with automatic start-up and reply from a distance in the internet”. In: Proceedings of the 10th WSEAS international conference on applied informatics and communications, and 3rd WSEAS international conference on Biomedical electronics and biomedical informatics, 2010, pp. 109–114.

[ZL12] Zomaya, A. Y.; Lee, Y. C. “Energy Efficient Distributed Computing Systems”. Wiley-IEEE Computer Society Pr, 2012, 1st ed..

[ZLLW12] Zhu, H.; Liu, Y.; Lu, K.; Wang, X. “Self-adaptive management of the sleep depths of idle nodes in large scale systems to balance between energy consumption and response times”. In: Proceedings of the 2012 IEEE 4th

International Conference on Cloud Computing Technology and Science (CloudCom), 2012, pp. 633–639.

[ZLOA13] Zhuang, H.; Liu, X.; Ou, Z.; Aberer, K. “Impact of instance seeking strategies on resource allocation in cloud data centers”. In: Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing, 2013, pp. 27–34.

[ZQR+07] Zong, Z.; Qin, X.; Ruan, X.; Bellam, K.; Nijim, M.; Alghamdi, M. “Energy-efficient scheduling for parallel applications running on heterogeneous clusters”. In: Proceedings of the 2007 International Conference on Parallel Processing, 2007, pp. 91–113.

[ZZA10] Zhu, Q.; Zhu, J.; Agrawal, G. “Power-aware consolidation of scientific workflows in virtualized environments”. In: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, 2010, pp. 1–12.