Universidade Federal Fluminense

ANDRÉ FELIPE DE ALMEIDA MONTEIRO

Quantum Virtual Machine: a dynamic approach for managing power and performance in virtualized clusters

Niterói

2017

Universidade Federal Fluminense

ANDRÉ FELIPE DE ALMEIDA MONTEIRO

Quantum Virtual Machine: a dynamic approach for managing power and performance in virtualized clusters

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial de qualificação para a obtenção do Grau de Doutor em Computação. Área de concentração: Redes e Sistemas Distribuídos e Paralelos

Orientador:
Orlando Loques

Niterói

2017

André Felipe de Almeida Monteiro

Quantum Virtual Machine: a dynamic approach for managing power and performance in virtualized clusters

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação. Área de concentração: Redes e Sistemas Distribuídos e Paralelos

Aprovada em Abril de 2017.

Niterói

2017

For the lovely Maui and Najara.

Acknowledgement

I would like to thank my supervisor Orlando Loques, and Julius Leite who was my supervisor for the first two years of the Doctorate, but retired in 2013. I will always be grateful for all their dedication and enthusiasm that they always shown during the Doctorate. They are the professional example that I would like to follow. A special thank to professor Alexandre Sztajnberg, who was my supervisor during my master's degree and show me the first steps in this long journey.

To my family and friends, specially my wife Najara and my daughter Luisa Maui, I would like to thank for all the support they gave me, mainly when they understood I need to be "offline"for some time.

Resumo

O elevado consumo de energia em ambientes de processamento de larga escala, como plataformas de computação em nuvem e data centers, mostra-se uma questão central nos dias atuais onde a sustentabilidade ambiental desses ambientes é uma premissa prioritária. Prover um gerenciamento energético eficiente é fundamental para reduzir os custos de operação e o impacto ambiental desses ambientes. Entretanto, esta é uma tarefa complexa, pois o gerenciamento energético deve lidar com diversos fatores como qualidade de serviço das aplicações suportadas, escalabilidade, heterogeneidade dos recursos de processamento, perfil energético e capacidade de processamento desses recursos, dentre outros. Este trabalho apresenta a Máquina Virtual Quântica, um modelo para gerenciamento de servidores virtuais de aplicação (Virtual Machines - VMs) em clusters de processamento. Além de prover economia de energia e garantir a qualidade de serviço das aplicações suportadas, nosso modelo tem escalabilidade linear e define um padrão de servidor virtual de processamento denominado de VM Quântica. Um conjunto de Máquinas Virtuais Quânticas constitui um Servidor Lógico Web (SLW), que opera de forma flexível ajustando seu consumo de energia e seu desempenho de acordo com a carga de requisições das aplicações. Conceitos de clone ágil e co-alocação de VMs e DVFS (Dynamic Voltage and Frequency Scaling) são utilizados em nosso modelo, viabilizando ações ágeis de reconfiguração do cluster e um controle fino de QoS das aplicações. Além disso, propomos uma modelagem em tempo de execução dos recursos de processamento, definindo dinamicamente o consumo de energia e a capacidade de processamento dos mesmos, dispensando assim a necessidade de uma análise prévia do ambiente de processamento. Os experimentos avaliam o nosso modelo por meio das métricas de consumo de energia e taxa de violação de QoS em comparação às políticas nativas do Linux para gerenciamento energético de servidores, e a modelos do estado da arte da área de gerenciamento energético de clusters. Os resultados demonstram que nosso modelo é capaz de economizar até 51.8% da energia consumida em um cluster que opera em sua capacidade máxima, com um impacto irrelevante no desempenho das aplicações.

Palavras-chave: Eficiência energética de servidores, gerenciamento de recursos, escalabilidade, virtualização, clusters de servidores.

Abstract

Power consumption in large-scale processing environments, such as cloud computing platforms and data centers, is a major issue nowadays, where the environmental sustainability of these environments is a priority premise. Efficient power management is critical to reduce the operating costs and the environmental impact of these environments. However, this is a complex task since the power management must consider several factors such as the QoS of the supported applications, scalability, heterogeneity of the processing resources, power and performance profiling of these resources, among others. This work presents a model for managing Virtual Machines (VMs) on server clusters. In addition to providing energy savings, our model has linear scalability and is independent of the processing platform. We define a default processing virtual server, named as Quantum Virtual Machine (QVM). A set of QVMs implement a Logical Web Server (LWS), which operates in a flexible manner, changing its performance and power consumption depending on the workload of the applications. Concepts of agile VM clone, co-allocation of VMs in the same core, and Dynamic Voltage and Frequency Scaling (DVFS) are used in the model, enabling rapid configuration actions and a fine-grained QoS control. In addition, we propose a runtime modeling of the processing resources, dynamically defining their power consumption and processing capacity, avoiding the need for a previous profiling of the processing environment. Experiments evaluate the effectiveness of the proposed model by means of power consumption reduction and QoS violations as compared to the Linux CPU governors and state-of-the-art energy-aware approaches based on optimization. The results show our model conserves up to 51.8% of the energy required by a cluster designed for peak workload scenario, with a negligible impact on the applications performance.

Keywords: Power-aware computing, resource management, scalability, virtualization, server clusters.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

API	:	Application Programming Interface;
CPU	:	Central Processing Unit;
DVFS	:	Dynamic Voltage and Frequency Scaling;
EDP	:	Energy Delay Product;
HTTP	:	Hypertext Transfer Protocol;
IaaS	:	Infrastructure as a Service;
LLCMS	:	Last Level Cache Miss per Second;
MIPS	:	Million Instructions per Second;
NFS	:	Network File System;
PMC	:	Performance Monitoring Counters;
QoS	:	Quality of Service;
QVM	:	Quantum Virtual Machine;
RPC	:	Remote Procedure Call;
SLA	:	Service Level Agreement;
LWS	:	Logical Web Server;
VAS	:	Virtual Application Server;
VCPU	:	Virtual Central Processing Unit;
VM	:	Virtual Machine;
XML	:	eXtensible Markup Language.

Sumário

Chapter 1

Introduction

Cloud computing plataforms and data centers are designed to provide large volumes of applications and services, which need to share the physical resources of the processing environment (servers, network equipment, energy, etc.). This concept of resource sharing is largely exploited by cloud computing providers, for instance Amazon EC2 [1] and Google Apps [23]. In general, these processing environments have a heterogeneous processing platform, due to the replacements of servers and hardware devices (e.g., maintenance, upgrades). As shown in [4], application servers usually operate between 10% and 50% of their maximum load. Another issue is the narrow dynamic power range of servers: even completely idle servers still consume about 70% of their peak power [18].

There are other crucial problems that arise from high power consumption by computing resources. Power is required to feed the cooling system operation. For each watt of power consumed by computing resources, an additional 0.5-1 W is required for the cooling system [62]. In addition, high power consumption by the infrastructure leads to substantial carbon dioxide (CO_2) emissions [24]. According to Gartner, the IT industry produces 2% of global CO_2 emission, which places it on par with the aviation industry [60]. More to the point, based on the trends from American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE), it has been estimated that by 2017 infrastructure and energy costs would contribute about 75%, whereas IT would contribute just 25% to the overall cost of operating a data center [5].

A virtualized application server cluster is implemented in processing environments by virtualization tools. This approach enables Virtual Machines (VMs) of various applications to run on the same physical server, sharing the processing resources such as CPU, memory, disk, etc. In scenarios of low utilization, due to low-load requests to the cluster, a VM consolidation process can be performed on a reduced number of physical servers.

This consolidation is made with VM migration procedures between physical servers in the cluster. This method enables a more rational use of environmental resources such as electricity, for instance, as it prevents servers from needlessly remaining powered on.

This reduced power consumption via consolidation occurs because most of the physical servers are oversized in order to withstand heavy request scenarios that are close to the threshold designed for the machines. Despite the beneficial effects of the VM consolidation process on energy savings, there is a restriction with regard to ensuring the Quality of Service (QoS) of the running applications in the cluster. Thus, it is necessary to find a balance in the consolidation process in order to make energy savings possible without compromising the application's QoS.

Another relevant technique capable of providing energy savings on physical servers is the DVFS (Dynamic Voltage and Frequency Scaling), as shown in [63] and [72]. This technique consists in adjusting the frequency and voltage of the processor at runtime, thereby adapting the processing capacity to the current workload demand. In other words, in low workloads, the processor's operating frequency can be decreased, thus reducing its processing capacity and its power consumption. As shown in our initial tests presented in Chapter 3, there is a linear relationship between the operation frequency of a processor and its performance. Moreover, the relationship between the operating frequency and the processor's power consumption is quadratic. Therefore, the trade-off between power and performance through DVFS must be addressed by a resource manager aimed at energy savings.

A paradigm related to the DVFS is to address the trade-off between power and performance through virtualized platforms composed of *big/small cores*. These platforms enable a mix of high performance and energy-efficient processors, in order to explore the energy savings and the performance constraints in processing environments. In this scenario, the set of *small cores* offer low but energy-efficient processing capacity while the *big cores* focus on high processing capacity at the expense of energy-efficiency. In other words, the *small cores* are designed for energy savings, while the *big cores* are indicated for high performance workloads. In general, the *big/small cores* platforms also implement the DVFS through resource managers in order to provide a fine-grained power and performance management according the workload. However, the range of available operating frequencies is more restricted in order to keep the particular features of each set (*small* and *big*) of cores.

As analyzed in [19], for certain types of applications (i.e., web applications) to use *small cores* is more interesting for the trade-off between power and performance. More to the point, this trade-off also depends on how well a load distribution strategy can map workloads onto the most appropriate core type. As shown in [59] some applications have a mix of processing phases (having heavy and light demands) and a dynamic scheduling between *big/small cores* combined with DVFS is required to manage the clusters in an energy-efficient manner. Thus, the state-of-the-art models aimed at power and performance management must present a high level of adaptiveness, in order to handle different types of applications and workloads. Moreover, these models also must address the heterogeneity of the processing environment, since homogeneous processing environments are uncommon in cloud computing platforms nowadays, as discussed earlier.

1.1 Research issues analyzed

Besides the management of the processing environment in a energy-efficient manner, the configuration of the active VMs in the cluster is a key factor for the applications performance. Thus, when a virtualized servers cluster is deployed, there are relevant issues that must be addressed by a resource manager. In particular, this thesis focus on the following research issues:

1. **The dependency between resource manager and processing platform**.
   Motivation: The main resource managers are dependent on the processing platform, since they must collect control data which are associated with a specific hardware family, such as IPS (Instructions Per Second), IPC (Instructions per Cycle), or cache miss rate, for instance. Moreover, in virtualized platforms, the management solutions implement new procedures to extend available functions in the virtualization tools, making solutions dependent on the virtualizer.
   Proposed goal: Our model should allow good results for power and performance management regardless of the processing platform.

2. **The need for power and performance profiling of the cluster's processing resources**.
   Motivation: Due to the heterogeneity of the processing platforms, which are composed of different families of processors and servers, is not trivial to define at runtime the power and performance profile of a processing resource. To this end, the most common approach is to perform previous profiling experiments in order to obtain the required information about the processing environment. However, it is imprac-

tical to stop the operation of data centers or cloud computing platforms to perform experiments. In addition to the need for suspending all the application services and tasks, which will have a major impact on the client's interface and violate Service Level Agreements (SLAs), several complex restoring procedures will be required after the experiments to resume the applications.

Proposed goal: Provide an accurate estimate of the cluster's processing resources through runtime modeling that avoids the need for previous power and performance profiling.

3. **The scalability of the virtualized server cluster**.

   Motivation: To define at runtime in which physical server the VM should be allocated and set the proper VM configuration can be a lengthy process when a large number of instances have to be analyzed. In general, the problem of allocating a VM in a cluster of physical servers and determining the VM configuration is modeled as a *bin packing* problem (NP-Hard), and solving it requires using an optimization process that has exponential complexity due to the combinatorial nature of the problem.

   Proposed goal: Achieve the scalability through an algorithm with linear complexity, which defines the reconfiguration actions on the virtualized cluster.

4. **The agility of the cluster reconfiguration actions**.

   Motivation: The reconfiguration actions defined by a resource manager must be performed in an agile manner, in order to guarantee the fine-grained QoS control required by the applications. To provide rapid reconfigurations actions, which is a key factor to obtain a scalable approach, the resource manager should simplify to the maximum the decision making process and the reconfiguration procedures. Thus, shorter control periods could be enabled and the cluster would be quickly adapted to handle peak workload scenarios.

   Proposed goal: Implement reconfiguration actions to quickly perform the resource management policies, enabling the cluster reconfiguration with a low latency in order to provide a fine-grained QoS control.

5. **The effectiveness of the DVFS and the big/small cores paradigm on the trade-off between power and performance**.

   Motivation: In general, the resource managers based on DVFS must also implement several extra procedures, such as advanced load balancing, control procedures using performance counters, etc., to provide a lower power consumption and a better performance. The use of multicore processors is the trend nowadays in modern processing platforms. Thus, the resource managers based on DVFS require the setting of the proper operating frequency for each core, which can be a complex analysis in large-scale processing environments. In contrast, the use of a *big/small cores* platform provides a relevant simplification of resource management, since each core is configured using its nominal value of operating frequency. Therefore, is possible to define sub-set of cores with similar operating frequencies and characteristics (energy savings or high performance), avoiding the complexity of the local DVFS policy for each core in scale-out platforms.

Proposed goal: Verify the energy-efficiency of a heterogeneous DVFS-based platform in comparison with a *big/small cores* processing environment.

Our work addresses the five issues mentioned above. We propose a model for simplifying the VM allocation process, using a canonical basis for all active VMs, called Quantum Virtual Machine (QVM). The QVM model aims to perform the power and performance management of server clusters in an energy-efficient manner. Associated with the model, two resource managers with different approaches for power and performance modeling of the cluster's processing resources are proposed and evaluated.

The first manager, called `Offline`, requires a preliminary cluster resource profile, which measures the power consumption and the processing capacity of each core in the cluster. The second manager (`Online`) performs the power and performance modeling of the cores during runtime, avoiding previous tests and analysis. Moreover, both approaches perform their policies based on a simple monitoring process, which needs periodically measure only the utilization rates from VMs and cores, avoiding metrics specifically related to the processor's configuration.

The resource managers are depicted in Fig. 1.1. As can be seen, both managers are based on the QVM paradigm. However, they differ in how the cluster's processing resources are modeled. Therefore, `Offline` and `Online` are responsible for the application of the premises described in our model, in order to provide a cluster management with high level of energy-efficiency.

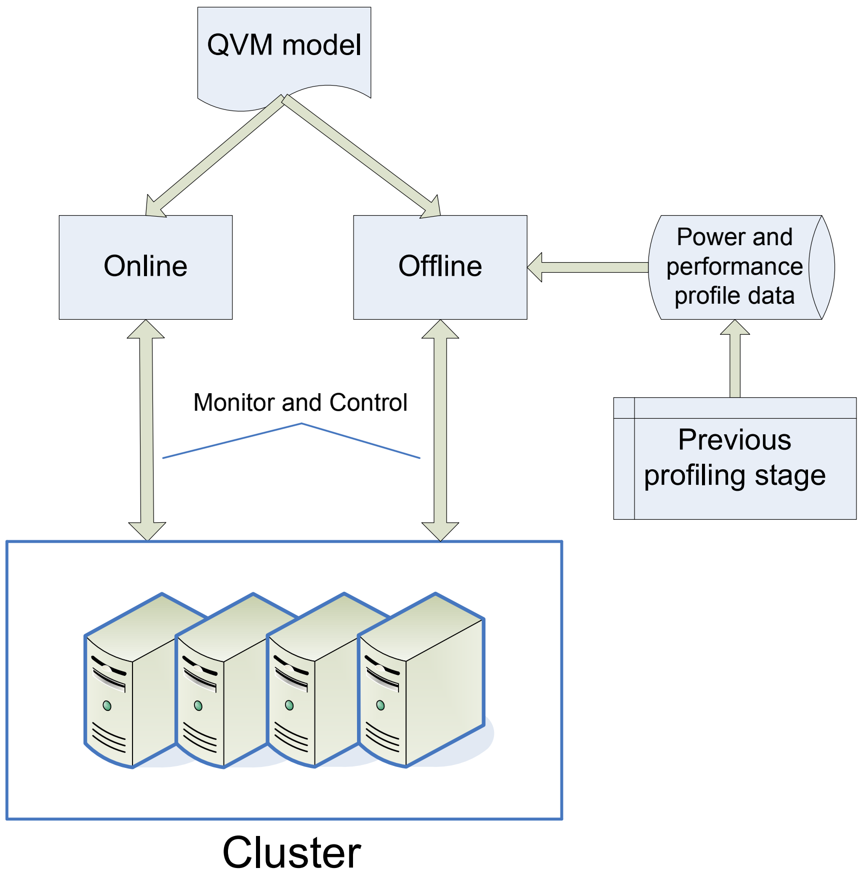


Figura 1.1: QVM model and resource managers

1.2 Thesis contributions

The contributions of this thesis can be divided into four categories: virtualization techniques, resource management in virtualized server clusters, runtime modeling of processing resources, besides the scalability and independence of the processing platform. The main contributions of this thesis are highlighted as follows:

- We address clusters with a high degree of virtualization, using state-of-the-art virtualization techniques, such as VM co-allocation in the same core and the VM agile clone;
- We propose and implement a background transfer in the agile clone process in order to decrease its latency;
- We present the QVM paradigm, which enables a novel VM cluster model and simplifies the VM allocation process in order to ensure scalability of our model;

- We achieve high energy savings while guarantee a negligible impact on the application's performance;
- We propose runtime profiling of the cluster's processing resources;
- The proposed resource managers are independent of the processing platform and require only a few simple measurements (utilization rates from the VMs and cores) to perform their policies.

1.3 Thesis organization

The core chapters of this thesis are derived from journal and conference papers, which were published by our group during the doctoral candidature. The relationship between these papers and the thesis, as well the text organization are described as follows:

Chapter 2 analyzes the main related works which address power and performance management in virtualized server clusters. The related works were divided according to their most common features. Moreover, we also highlight the main contributions of our proposed model in comparison with the analyzed works.

Chapter 3 presents the main concepts and technologies used in our model, and we describe the initial experiments performed to ratify these concepts. We also present and discuss the main virtualization techniques implemented in our approach. In addition, we analyze the trade-off between power and performance of a core when the DVFS is used. These preliminary tests were conducted on the first stage of our research, where the assumptions used in our model were validated empirically. The results were detailed and analyzed in [55], and the conclusions lead to the main ideas used in the development of our proposed model.

Chapter 4 describes the proposed model, and we also present a detailed analysis of our approaches for resource management. The first version of our model was presented in [51], which defined a single VAS (Virtual Application Server) standard, called Quantum VAS because of its reduced processing capacity and use of a VAS co-allocation process in the same core processor. Then, in [53] we used the same concepts from previous work, but we focused on the evaluation of more aggressive reconfiguration actions in the cluster, in order to maximize the energy savings and evaluate their impact on applications QoS.

The QVM concept was presented in [52]. A set of QVMs implement a Logical Web Server (LWS), which operates in a flexible manner, changing its performance and power consumption depending on the workload of the applications. Concepts of agile VAS clone, co-allocation of VAS in the same core, and Dynamic Voltage and Frequency Scaling (DVFS) are used in the model, enabling rapid configuration actions and a fine-grained QoS control. Finally, in [54] we present the runtime modeling of the cluster's processing resources, which avoids the need for previous power and performance profiling. Thus, the model proposed in this thesis provides a detailed discussion on how the concepts used in each previous work were combined. Besides that, we analyze the main issues related to each concept with focus on the trade-off between power and performance.

Chapter 5 presents a performance evaluation of the proposed model. The two resource managers implemented in our model were compared with Linux CPU governors, state-of-the-art power-aware approaches based on optimization, and resource modeling based on Performance Monitoring Counters (PMC). The effectiveness of each approach can be observed in the behavior of the applications supported by the VMs, and in the power consumption of the cluster. To this end, all the approaches were set to avoid the degradation of the applications performance, and to manage the cluster's resources in an energy-efficient manner.

In order to observe the efficiency of the evaluated approaches in different processing platforms, we used two different test scenarios. The first scenario consists of a `Big/Small Cores` platform and does not use the DVFS to dynamically change the operating frequency of the cores. The second test scenario consists of a set of heterogeneous cores with different processing capacities and power consumption, in which the DVFS is used by all the evaluated approaches. These platforms were used in [55] to manage a virtualized server cluster composed of three physical servers that together have a total of 176 cores, enabling the scalability analysis with a suitable setting.

Chapter 6 discusses our conclusions and indicates some future work. Besides verifying along the thesis, the five research issues pointed out earlier, Chapter 6 (Conclusions) summarizes how each issue was addressed by our model, and highlights their respective proposed solutions.

Chapter 2

Related Work

The issue of energy-efficiency in scale-out data centers and cloud computing platforms is a major trend nowadays and many different works can be found in the literature. In this chapter we analyze the state-of-the-art approaches which address the power management in processing environments. In order to provide a better discussion of these works, we divide them according to their most common features.

2.1 Energy-efficient processing environments

As pointed earlier in Chapter 1, cloud computing environments usually are composed by heterogeneous processing resources due to the replacement of servers and hardware devices. Even when the processing platform was initially deployed with homogeneous servers, the maintenance and upgrade activities over time bring heterogeneity into the environment. To provide energy savings in heterogeneous platform, the works described in [30] and [31] indicate the use of low processing capacity and low power consumption processors, called *small cores* or *wimpy cores* in server clusters. These processors are not designed for high performance processing but for energy savings, enabling low power consumption due their simpler hardware configuration and lower operating frequencies. As described in [19], for certain types of workloads such as web applications, for example, the *small cores* is the best choice to address the trade-off between power consumption and performance. Otherwise, for applications that require a heavy processing load, an environment composed of multicore processors with high processing capacity (*big cores* or *brawny cores*) may be the most suitable.

Several works present power awareness approaches based on DVFS, as described in [47], [33], [37] and [44]. The use of multicore processors nowadays enables the setting of the proper operating frequency for each core, which provides local and global power management of the processing resources. However, it is important to notice that a local DVFS policy has upper and lower operating limits. The lower limit is associated with the minimum occupancy in the lowest operating frequency of the core. This occurs when it is not possible to reduce the processing capacity and the power consumpution of the core through a lower operating frequency. The upper limit of DVFS is the maximum occupancy using the highest frequency of the core, which ratifies the saturation point of the core's processing capacity. These lower and upper limits indicate to the global management the need for activating a new core, or the possibility to deactivate an operating core.

As DVFS reduces the number of instructions that a processor can handle in a given amount of time, it increases the run-time of program segments that are significantly CPU intensive. Thus, this scenario creates a challenge of providing the optimal power/performance management, which has been extensively investigated in recent years. The major impacts on the applications performance through the use of DVFS are analyzed in [31], [13] and [65]. These works argue that the DVFS bounds are not always able to assure the performance required for the supported applications, in addition to not always reaching satisfactory levels of energy savings.

Although the implementation of the DVFS may seem to be straightforward, real-world systems raise many complexities that have to be taken into account. Since the complex architectures of modern CPUs (advanced pipelines, multilevel cache, etc.), the prediction of the required operating frequency that will meet the application performance requirements is not simple. From this perspective, a key advantage of our model is to provide significant energy savings with negligible impact on the applications performance as shown in our performance evaluation. To verify this feature in our model and to analyze the effectiveness of the DVFS, we performed experiments using two different platforms, one composed by heterogeneous cores using DVFS, and other composed by *big/small* cores without DVFS. The results described in Chapter 5 - Performance Evaluation ratify the lack of DVFS regarding the applications performance and energy savings in comparison with the *big/small cores* platform.

2.2 Approaches based on virtualization

Among the benefits of virtualization are improved fault and performance isolation between applications sharing the same physical server, and the ability to relatively easily move VMs from one physical host to another using the VM migration procedure. Besides that, the virtualization process enables a high level of abstraction, providing a wide support for hardware and software heterogeneity. The ability to migrate VMs at runtime enables a dynamic VM consolidation policy aimed for energy savings on processing platforms. However, workload consolidation is a non-trivial problem since aggressive consolidation may lead to performance degradation of applications. Therefore, the VM consolidation process is typically constrained by QoS requirements.

Several power-aware approaches use virtualization techniques, as shown in [45] and [73], because of the potential for energy savings through the consolidation of VMs. Although these works provide energy-efficient server clusters through virtualization, they do not enable resource management with rapid reconfiguration actions and a fine-grained QoS control. To perform the VM consolidation, the cited works use the live migration procedure to migrate VMs between the physical servers, which has high latency. In our model we use a VM agile clone procedure, which performs a faster migration of VMs without harming the performance of the applications, as shown in the analysis presented in Chapter 3.

The works described in [50] and [7], which implement a server cluster through virtualization, also use the live migration process of the virtualizer to consolidate VMs in a physical server. In these works, the proposed resource managers consolidate the VMs in a reduced number of servers, allowing to turn-off the other ones for energy savings. As the control granularity determined for those management schemes is high (3-5 minutes), the latency of the live migration does not seem relevant. However, to perform a fine-grained control process, the live migration becomes an relevant problem and the use of more agile procedures for VM migration between the servers cluster is required.

Another approach frequently observed in state-of-the-art models designed for energy-efficient virtualized server clusters regards determining the best configuration of the active VMs by means of an optimization process, based on the *bin packing* problem, as described in [6], [74] and [39]. Thus, the optimization problem is modeled to minimize the power consumption of the cluster, restricted to the capacities of the physical resources (number of active cores, CPU operating frequency, amount of available memory, etc.) and to the applications QoS.

The work described in [20] studied the problem of allocating an available power budget to servers in a heterogeneous server cluster to minimize the average response time of web applications. The authors investigate how the operating frequency scaling techniques affect power consumption. They conducted experiments applying DVFS for web application workloads. The results showed a quadratic power-to-frequency relationship for DVFS techniques. Given the relationship between power and performance, the cited work verifies the problem of finding the optimal power allocation as a problem of determining the optimal frequencies of the DVFS for each server, while minimizing the mean response time. In order to investigate the effect of different factors on the mean response time a queueing model was implemented, which allows prediction of the mean response time as a function of the power-to-frequency relationship, arrival rate, peak power budget, and so on. The model allows determining the optimal power allocation for every configuration of the above factors, but the proposed solution is highly related to the processing platform used in the experiments.

The model proposed in [58] uses optimization based on mixed interger linear programming to determine the cluster configuration that minimizes power consumption, restricted to non violation of the application's QoS. The DVFS, server on/off and VM live migration are used to perform the reconfiguration actions. It is important to notice that in the optimization approaches, scalability can be a bottleneck due to exponential complexity of the *bin packing* problem (NP-Hard). This bottleneck can be addressed by the use of black-box heuristics available in optimization solvers, in order to reduce the computational effort to define the best cluster configuration. However, significant gaps may exist between the quality of the solutions obtained by the heuristics and the optimal solution, implying smaller energy savings.

The schedulers of the virtualization tools was also analyzed in the literature. As the scheduling process is responsible for determing which VM should run, it has a relevant role in the application's performance. A heterogeneous multicore environment is used in [74] and [21] to evaluate the VM allocation problem aimed to minimize the power consumption. The cited works propose VM schedulers to consider the heterogeneity of the processing environment, as opposed to the default scheduler used by the most popular virtualization tools, which does not take this aspect into consideration. However, the models do not explore concepts that could improve their performance, such as the co-allocation and agile cloning of VMs supported by our model.

2.3 VM power and performance modeling

Several state-of-the-art approaches for modeling power and performance of VMs present utilization-based models. The work presented in [18] evaluates the relationship between the VCPU (Virtual CPU) utilization and total power consumption by a server. The idea behind the proposed model is that power consumed by a server grows linearly with the growth of the VCPU utilization of the VMs, from the power consumed in the idle state up to the power consumed when all the VMs are fully utilized.

The modeling proposed in [49] and [8] are based on a simple assumption that the power consumption of a physical processing resource is linear to its CPU utilization rate. The authors consider that when a physical server hosts VMs, its overall power consumption can be calculated as the sum of the power consumption of the VMs. These models consider that there is a static power consumption equally shared by all VMs, and a dynamic power consumption which is proportional to the VM performance. In other words, the more a VM uses the CPU for processing its workload, the greater its power consumption is. As the utilization-based models address the individual performance and power consumption of each VM, they have to obtain some previous information about the processors where the VMs are allocated. The works described in [45] and [32] point out that the need for previous profiling tests occurs due to the specific hardware configuration of each processor, which is related to its design and manufacturing process.

The work presented in [46] analyzes the trade-off between power consumption and performance in the VM consolidation process. The paper refers to an upper bound in the VM consolidation, in order to maintain the performance of the applications under the Service Level Agreement (SLA) specifications. The main goal is to avoid a forced competition for physical resources (memory and CPU time) between all the active VMs, which would lead to a contention scenario for applications and a possible QoS degradation. A previous power and performance profiling of the processing platform is required in order to provide the cluster modeling. However, as discussed earlier, the need for previous experiments is a bottleneck in large-scale data centers and cloud computing platforms. We also used threshold values in our model to prevent *idle power* and contention of requests, which leads to QoS degradation. However, we propose a runtime approach to model the cluster's physical resources, which eliminates the need for offline resource profiling.

The use of Performance Monitoring Counters (PMCs) as proxies for power and performance measurements is widely observed in the literature. The PMCs provide a number of measurable performance metrics, which enables the measurement of the system state of any given application.The main advantage of PMCs approaches in comparison with utilization-based models is their capability to avoid the previous power and performance profiling. Therefore, as the resource manager proposed in this thesis, called `QVM Online` and described in Chapter 4, the PMC-based approach is able to perform the resource modeling at runtime. The modeling of the cluster's processing resources is based on performance counters, which have been widely supported in modern multicore processors. These counters will accumulate all system-level events that were triggered by different components. By accessing these monitoring counters, such as IPS (instructions Per Second) or cache miss rate, for instance, a power and performance modeling can be implemented, as described in [11] and [61].

In the work described in [15] the authors show that although regression models based on VCPU utilization are able to provide reasonable prediction accuracy for CPU bound workloads, they tend to be considerably inaccurate for prediction of power consumption caused by I/O bound applications. A power modeling methodology based on Gaussian is proposed by the cited work, which uses several models that predicts power consumption by a physical machine running multiple virtual machine instances. To perform predictions, in addition to the VCPU utilization, the model relies on runtime workload characteristics, such as the number of Instructions per Cycle (IPC) and the number of cache misses.

In [10] the virtualization technology is used with PMC-based power modeling techniques to derive power consumption estimates of the CPU and memory at per-VM level. The cited work also analyzes the effects of DVFS on such scenario, and proposes a simple methodology to perform per-VM power accounting on current multicore processors which leverages the PMC-based power models. On the other hand, this feature requires a comprehensive knowledge of the particular architecture and the design of complex measurements procedures. Thus, this analysis introduces a relevant complexity for the modeling and implementation stages.

The need for ad-hoc tecnhiques in order to accurately measure the performance counters in virtualized environments is addressed in [56] and [76]. Because of the interaction of virtual machines with multiple underlying software and hardware layers, the analysis of the application's performance running in virtualized environments has been quite complex. This occurs because the monitoring activities through daemons in different domains need to collect several hardware events individually at the same time, and some of these events are not available in the virtualization level since they are related to the physical resources. As our model requires only the utilization rates from the VMs and cores to perform its policies, it becomes independent from the processing platform and also from ad-hoc procedures for monitoring the running VMs.

2.4 Scalability

A common technique to address scalability is to design the cluster in a hierarchical topology, segmenting the whole cluster in small parts where a local resource manager performs a local optimization process. Although this option is effective to reduce the time spent to define a good solution, the energy savings could be decreased in large-scale clusters scenarios, since a good local configuration of a cluster segment does not guarantee that a good global configuration is achieved. Our group's previous work [67] proposes a hierarchical structure among the cluster components to reduce the complexity of the reconfiguration actions. The goal is to avoid an optimization process using a resource configuration algorithm with linear complexity to provide energy savings. The servers were divided into segments, where an intermediate controller element was inserted for resource management and load distribution within each segment.

In [48] a power management is proposed to group similar threads (with common execution phases) on the appropriate core, regarding the core power consumption and performance. The cited work points that grouping similar applications on set of cores enables a better management of the processing resources in scale-out clusters, since a particular analysis of each application behavior is unpractical in large processing environments with thousand of running applications simultaneously. Therefore, each thread set is assigned to most appropriate cores according to the threads phase and type of workload.

However, the both mentioned works do not use virtualization, and the balance between local and global configuration is not considered. In our model, we address scalability without the need for cluster segmentation or black-box heuristics. The scalability is achieved through a new VM cluster modeling, which provides a substantial simplification of the VM allocation problem in the processing environment and enables the use of an algorithm with linear complexity in our cluster reconfiguration procedures.

Chapter 3

Basic Concepts and Initial Experiments

In this chapter we present and discuss the basic concepts used in our model and the initial experiments performed to ratify these concepts. First, we present the main virtualization techniques implemented in our approach. Then, we analyze the trade-off between power and performance of a core when its operating frequency is adjusted through the DVFS.

3.1 VM agile clone

Virtual machine consolidation is an effective way to minimize the power consumption of processing environments. However, identifying the right time to trigger migration is crucial specially when the host server is overloaded. The work described in [22] presents an energy-efficient reactive migration controller that identifies situations in which the hosts are overloaded or underloaded. The overload and underload detection is defined when the processor and memory utilization goes beyond or under a given threshold rate, respectively. The same approach is used in [7] and the effects of these two thresholds on the overall processing platform power consumption and QoS violations are analyzed. As shown in the cited works, rates of 30% and 80% are efficient underload and overload thresholds considering the total power consumption and application QoS.

In order to ensure a certain level of QoS, is also important that the virtualizer should guarantee the serving of the applications requests during the migration process. The virtualization tools provide a native VM live migration mechanism. This feature enables a VM to migrate between two physical servers in a cluster without interrupting the services provided by the VM, by first transferring its memory image and then the current state of its VCPU between the physical servers. The traditional live migration process is called pre-copy, related to the transfer of memory pages. When transferring the memory image some pages may be updated, since the VM is still running in the source server so that the service provided by the VM is not interrupted. Then, at the end of the transfer the copy of the pages that were changed in the source server during migration is also required. Such altered pages are called “dirty pages”. The entire transfer process is complete when the copy of "dirty pages"from the source server is concluded, ensuring that there was no disruption of the applications that are running on the migrated VM, followed by the deactivation of the VM in the source server. Lastly, a copy is made of the contents of the VCPU and VM registers in the source to the destination server, enabling the activation of the VM in the destination server with the same context.

In a server cluster, where a set of VMs needs to respond to requests sent by clients, the VM performance is impacted during the live migration process due to the transfer of “dirty pages”. As the VM will be operational in the destination server only after copying these remaining pages, and the VM in the source server undergoes a cut state precisely to define the ”dirty pages” to be transferred, there is a contention period of the requests met by the VM. These request restraints are directly reflected in the applications response time, as pointed out in [45].

The VM agile clone process, described in [28] and [29], differs from traditional live migration in terms of how memory pages are copied between source and destination servers, as shown in Fig. 3.1. Here, the first actions in agile clone regard copying the content of VCPU and VM registers, and transferring this content between the source and destination server. Next, a VM clone is immediately activated in the destination server, even without transferring any memory pages. When the VM clone searches for a memory page not transferred, its operation is momentarily stopped and the transfer of the requested page between source and destination server is made on demand, thus resuming the processing of the VM clone. This migration process is termed as post-copy, because the memory pages are dynamically transferred to the destination server when necessary. This on-demand transfer is performed until all VM memory pages are effectively transferred, thus enabling to deactivate the VM in the source server.

The post-copy process used in our model is similar to the approach described in [40]. In addition to the technique used in the cited work, we implemented a procedure in the virtualization layer to improve the performance of the VM agile clone in order to reduce its latency. We designed a background transfer of memory pages not yet sent, so that the transfer page operation is not completely on-demand. This background transfer is performed to prevent an overload on the data network when many agile clone processes

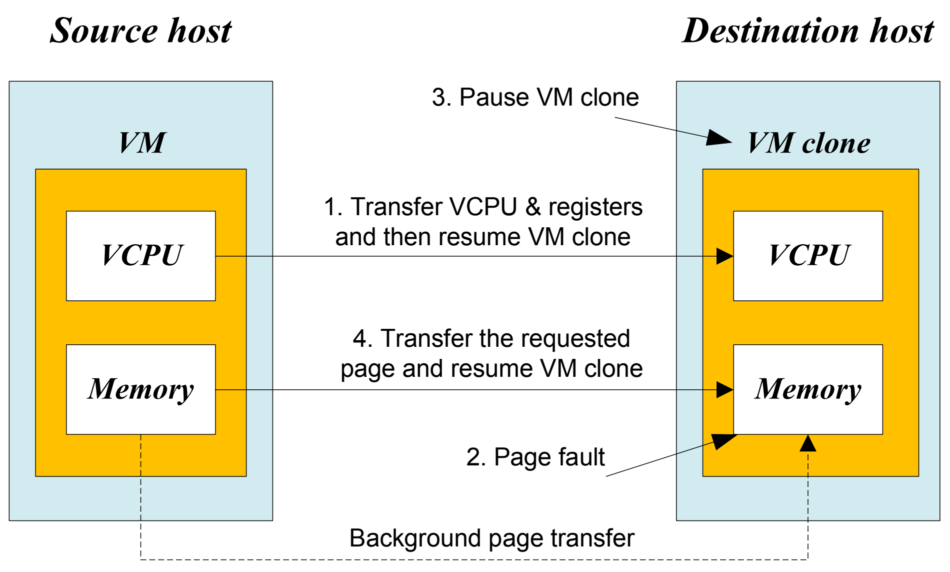


Figura 3.1: VM agile clone process

are running in a short period of time.

A set of initial tests were carried out using the `Web Serving` workload of the CloudSuite 2.0 [14] benchmark, which implements a large scale web system. This benchmark consists of three main elements: a web server, a database backend and a client to emulate real world accesses to the web server, each running on a separate machine. These tests were performed to verify how the migration procedures impact on the VMs performance, providing an experimental evaluation between the live migration and the agile clone with our proposed background transfer. To this end, we used the same server (Intel I7 2.5 GHz, 8 GB of RAM and 512GB of HD) in the live migration and agile clone experiments, and the VMs were configured with 512 MB of RAM and 2 GB of HD. The same benchmark was used to evaluate the overhead of the co-allocation process on the application performance. The co-allocation evaluation is presented in Section 3.2.

Fig. 3.2 shows the average response time when a constant amount (120 req/s) of web requests is handled by a single VM. When there are no procedures of live migration or agile clone, the average response time within one second is about 0.22 ms. Then, at the instant $t = 6s$ a VM migration to another physical server is started. The VM was activated on the destination server at $t = 6.2s$ through the agile clone, and the last memory page was transferred at $t = 8.7s$. However, when the live migration is performed, the VM was activated on the new server only at $t = 10.7s$. Moreover, as depicted in the graphic the impact on the application response time during the migration process is much higher in the live migration in comparison with the agile clone.

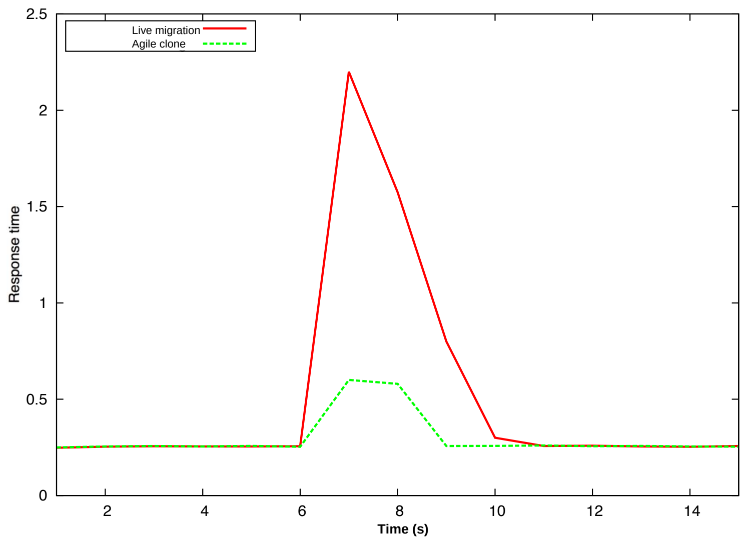


Figura 3.2: Single VM performance - Live migration vs. Agile clone with background transfer

The next experiment evaluates how the number of simultaneous VMs migration procedures impact on the VMs performance. As scale-out processing environments have a great number of physical servers and active VMs, several migration procedures could be performed at the same time by the resource manager when a peak workload scenario occurs, for instance. As for the first experiment, the metric used herein is the average response time of the web requests served by the VMs. To better visualize the results, the response time was measured only during the migration procedures. For all the test scenarios, there are twenty active VMs serving the web requests. Fig. 3.3 presents the measured values of response time when a different number of VMs are migrated through the live migration and agile clone.

As the previous test, the average response time is about 0.22 ms when no migration procedures were performed. Thus, as can be seen in Fig. 3.3, when only two VMs are migrated through the agile clone the impact on the response time is negligible. However, even for this low number of migrated VMs, the live migration presents a relevant impact on the application response time. For all the test cases, the impact on the VMs performance by the live migration is higher than agile clone. This occurs due to the significant delay in the requests when the live migration of a single VM is performed, as ratified earlier in Fig. 3.2. Therefore, when a great number of VMs are migrated through the live migration, the average response time of the set of active VMs is affected. On the other hand, as the agile clone causes a much lower delay in the requests, the average performance of the active VMs does not present a relevant variation.

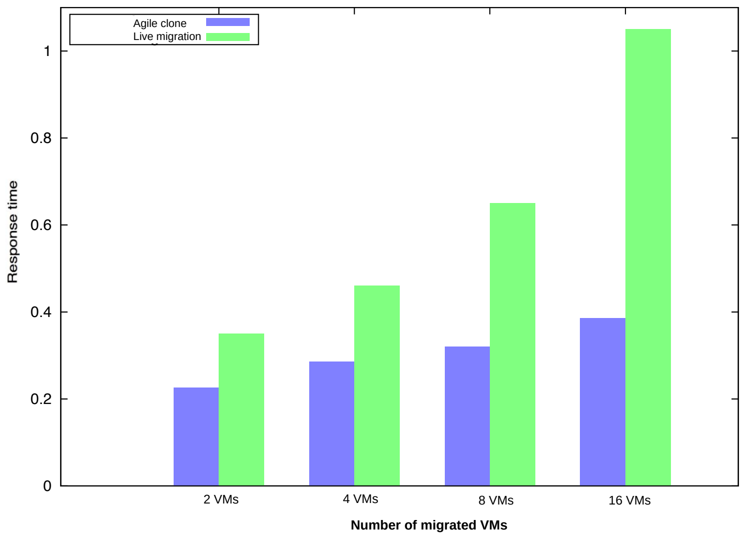
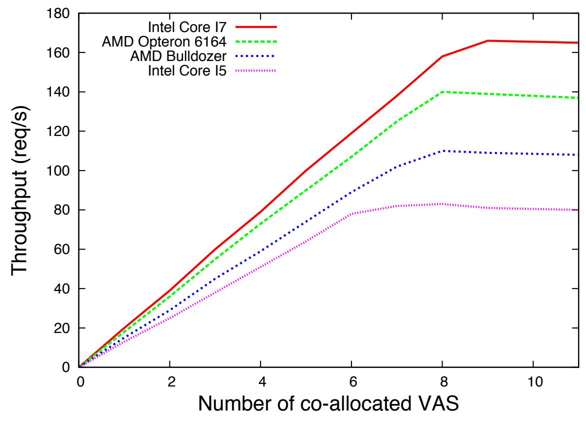


Figura 3.3: Set of VMs performance - Live migration vs. Agile clone with background transfer

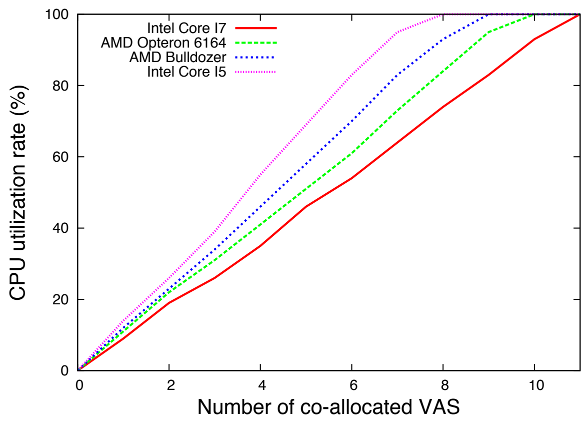
3.2 Co-allocation of VMs

The co-allocation of VMs on the same core uses the same concept of the VM consolidation: exploit to the maximum the resources sharing, deactivating the processing resources that are not processing data. Therefore, to minimize the number of active cores, achieving relevant energy savings, the co-location of VMs on the same core appears to be an effective strategy. The co-allocation process aims to reduce the waste of power, since a set of VMs are running on the same core, minimizing the time spent in the idle state and consequently the core *idle power*. However, as the VMs allocated on the same core share the same physical resources, such as private cache memory, the impact on the performance of co-allocated VMs needs to be checked. To this end, the works described in [19] and [34] indicate that for large scale systems the amount of cache memory available to the core does not have a relevant role, since the volume of memory processed is much greater than the capacity of the cache memory.

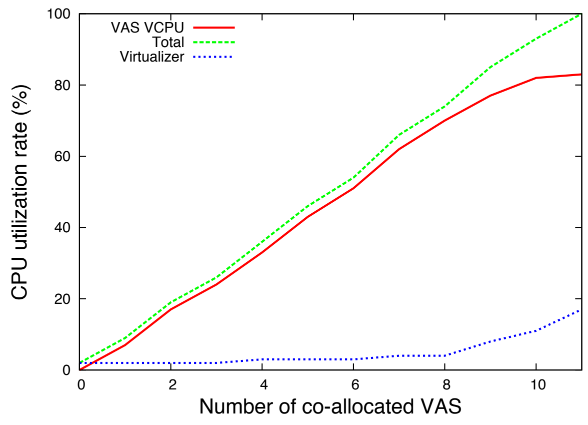
The goal of the co-allocation experiments performed with the CloudSuite benchmark was to define a saturation point of a core. In other words, how many VMs a core can co-allocate without performance degradation of its hosted VMs. Fig. 3.4-a shows the saturation point of the co-allocation occurs when the CPU utilization rate is about 80%. At this point, the requests' throughput does not increase when more VMs are activated. Furthermore, Fig. 3.4-b shows that the physical CPU utilization rate has a direct linear relationship to the number of co-allocated VMs. Then, Fig. 3.4-c shows that the average VCPU (Virtual CPU) utilization of the set of active VMs and the physical CPU utilization



(a) Throughput



(b) CPU utilization rate



(c) Useful CPU utilization

Figura 3.4: Co-allocation performance evaluation

rates are very similar up to the value of 80% ratifying the occurrence of the saturation point when a great number of VMs are co-allocated. From this point we can see that CPU utilization rate assigned to the virtualizer grows significantly, decreasing the processing window of requests by the set of VMs.

Finally, the scheduling strategy for co-allocated VMs should also be noted, in order to prevent a VM from having its performance degraded due to the competition for processing time. By using virtualization, the VM scheduling is assumed by the virtualizer based on its own policies, which can use round-robin schemes, time bounds to access the CPU, etc. In our work, the Xen virtualizer [3] is used for the model implementation and performance evaluation (Chapter 5). The Xen natively implements a policy based on credits to ensure equal access to the CPU by the VMs. Thus, every time a VM assumes the CPU, the VM spends part of its credits and has a lower priority for its next access. The goal is to allow other VMs with more credits to process their data. As the scheduling policy analysis is not part of the scope in this thesis, the original Xen scheduling was used in the model implementation.

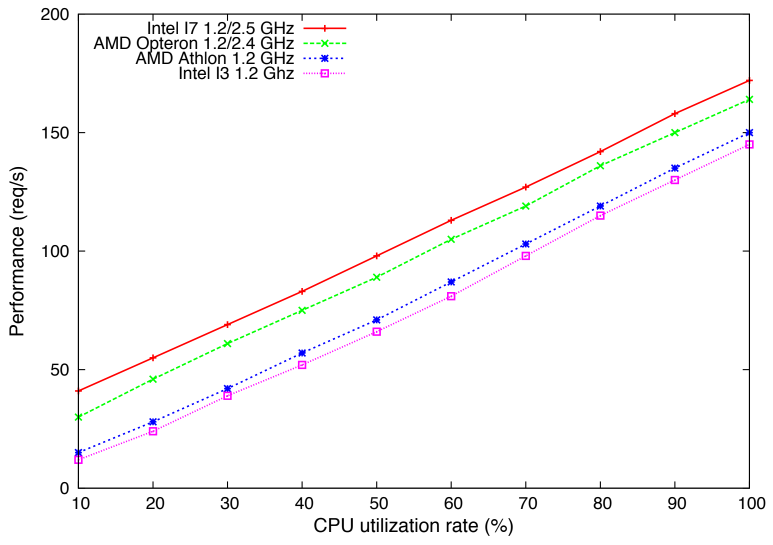
3.3 Big/Small cores and DVFS

To verify the impact of the DVFS on the application performance and on the power consumption of a core, we also performed tests using the CloudSuite benchmark. Additionally, we analyzed the trade-off between power and performance when *big/small cores* are used to support applications, as discussed in Chapter 2. As shown in [41], [43] and [35], a reduced operating frequency leads to a lower power consumption. However, this reduced operating frequency also leads to an increase of the request's turnaround time. Therefore, the energy consumption (*power* x *time*) of a request under different operating frequencies must be taken into account by the resource manager.

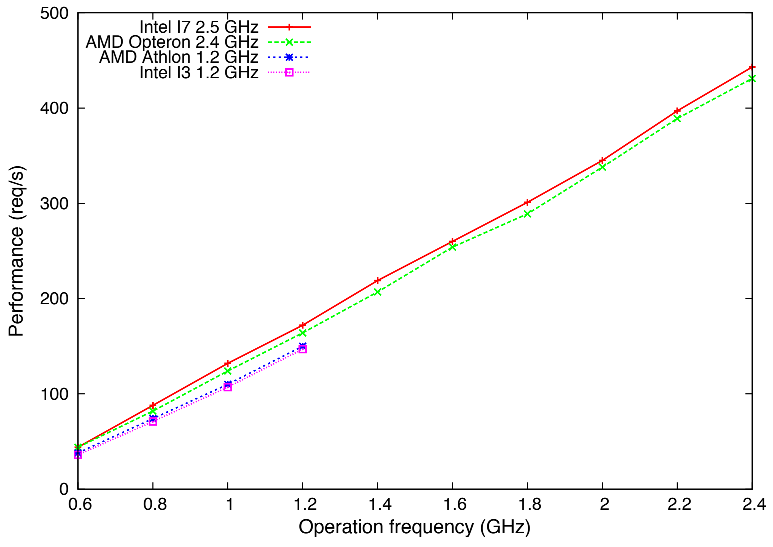
Although relevant works also analyze the trade-off between power consumption and performance of a processor, as shown in [58], [17], and [38], we performed our own tests in order to ratify the results verified in the literature, and to provide new insights in our work. To this end, we used four different processors to implement a *big/small cores* testbed. Therefore, we chose the AMD Opteron 2.4 GHz and the Intel I7 2.5 GHz as *big cores*, and the AMD Athlon 1.2 GHz and the Intel I3 1.2GHz as *small cores*.

Fig. 3.5-a shows the linear relationship between the number of completed requests and the utilization rate. We measured the performance of the cores in terms of the number of requests per second (req/s) they can handle at a given target utilization rate, for a constant operating frequency. Thus, for the selected operating frequency the number of requests is set to keep each core with its maximum (100%) utilization rate. In this case, we set the operating frequency of all cores to the greatest operating frequency value of the *small core* set (1.2 GHz), in order to provide a fair comparison between them. As seen in Fig. 3.5-b, the performance of a core is also proportional to its operating frequency, and there is a linear relationship between these two variables.

We also analyzed the relationship between the power consumed by a given core and its utilization rate for a constant operating frequency (1.2 GHz). As shown in Fig. 3.6-a, the relationship between these two variables is linear when the operating frequency is kept constant. Finally, Fig. 3.6-b shows that the power consumed by a core is proportional (in a quadratic form) to its operating frequency at a constant utilization rate. In all power measurements, we used the WattsUP Pro [75] power meter to collect the power consumption of the whole server. The WattsUp Pro displays the wattage (power) being consumed by all the machines which are connected to it. However, we are interested in the power consumption of cores rather than servers. Thus, first we measured the



(a) Performance vs. CPU utilization @1.2GHz

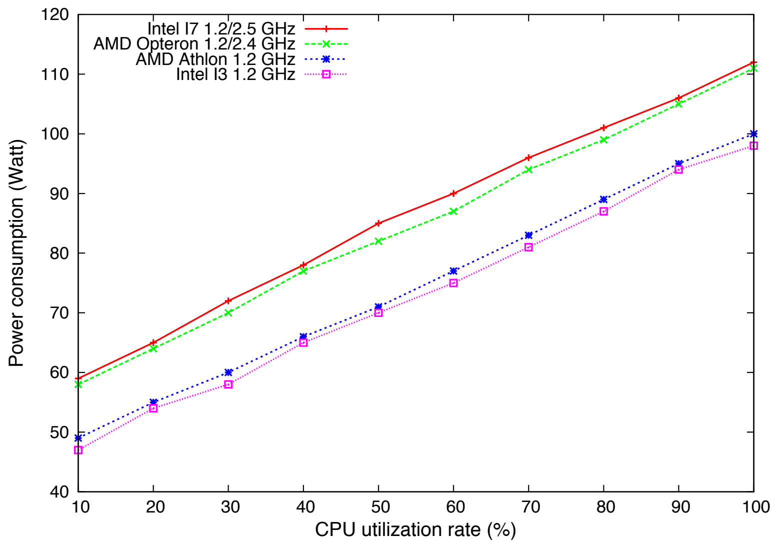


(b) Performance vs. Frequency @100% utilization

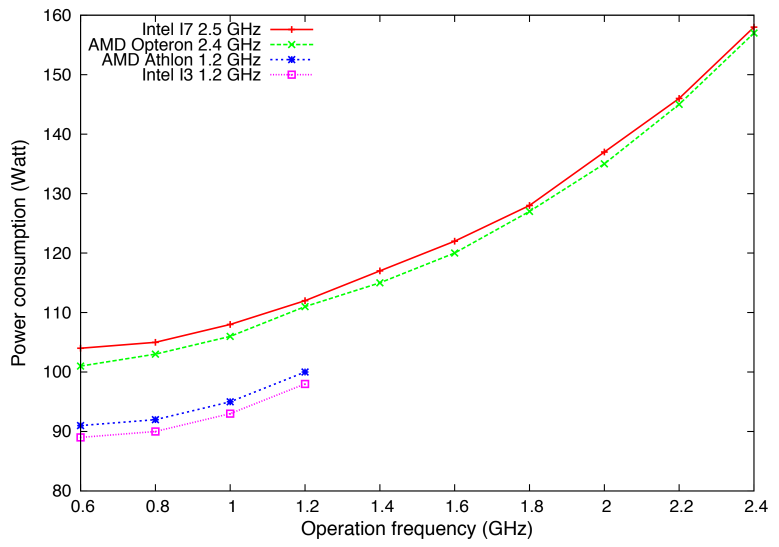
Figura 3.5: Performance on DVFS

power consumption of the entire server setting all the cores to the idle state. Then, the workload was assigned to a selected core and the power consumption was measured again. The difference between these two values can be considered as the individual power consumption of the selected core.

According to these initial tests, cores with the same operating frequency have different behaviors in relation to their power consumption and performance. This occurs due to the hardware configuration of each processor, which is related to its design and internal organization. Generally, processors designed to obtain high performance (*big cores*) have more complex internal components, such as advanced pipelines, bigger ALU memory, more auxiliary registers, etc. This means there is an extra power consumption arising



(a) Power vs. CPU utilization



(b) Power vs. Frequency

Figura 3.6: Power consumption on DVFS

from these components, which reflects higher power consumption of the whole core, as verified in [25] and [12].

Unlike the *big cores*, the *small cores* are not designed for high performance processing but for energy savings, enabling low power consumption due its simpler hardware configuration. This point is evidenced in Fig. 3.6-a, where the *big cores* have significantly greater power consumption than the *small cores*, even with equal operating frequency (1.2 GHz). However, it should be noticed that the *small cores* used in our experiments are not modern processors designed for energy savings, but traditional processors with low processing capacity due to the limitations imposed by our available testbed. Thus, the power consumed by the simulated *small cores* (Fig. 3.6-a and 3.6-b) should be even lower if real *small cores* could be used in our experiments.

As the operating frequency has a quadratic impact on the power consumption of a core, wich as ratified by our power experiment presented in Fig. 3.6-b, we can conclude that it should be more efficient to achieve better performance through a high CPU utilization rate than through a high operating frequency. Therefore, to achieve relevant energy savings, we might keep all cores with high utilization rates and set their operating frequency as low as possible. Although using a lower frequency leads to a longer processing time, the opposite strategy for energy savings (known as race-to-idle) does not work with all workloads, as shown in [2] and [27]. These cited works indicate that for energy savings in web applications, it is more efficient to have more cores at lower frequency than few cores at higher frequency.

So, based on the above described relationships, we present the details of the proposed model and its resource managers in the next chapter.

Chapter 4

The Proposed Model

Our model addresses the issue of modeling the dynamic power and performance processing resources in virtualized servers clusters. The goal is to model the cluster's resources appropriately in order to allow their efficient utilization and reduce their power consumption by reducing the *idle power*, while observing the QoS performance constraints. First, we present the QVM element, and we describe how it is related to the simplification of resource management achieved through our approach.

The proposed resource managers are responsible for applying the premises described in our model, in order to manage the cluster in an energy-efficient manner. Therefore, they should be based on the three **Definitions** presented in the follow section, using the QVM paradigm and the normalization of the VM's performance. Addotionally, as environment can be composed of heterogeneous servers and processors, the resource managers have to obtain information about power consumption and processing capacity of each processing resource. This information is crucial to enable energy savings and the fulfillment of the application's QoS restrictions.

The two implemented resource managers differ in how the cluster's processing resources are modeled. The `Offline` approach performs a previous power and performance profiling of the processing platform, evaluating the power consumption and the processing capacity of each processing resource available in the cluster. Despite the reservations discussed earlier about the complexity for freezing the cluster operation, the a priori profiling is widely used by resource managers found in literature. The main reason is the detailed information about the cluster's resources that can be obtained through the profiling stage.

To avoid the need for profiling, we present the `Online` approach, which is performed at runtime in order to continuously evaluate the cluster's resources using control periods. The dynamic modeling enables our runtime approach to achieve a high level of adaptiveness, making it independent from the processing platform and also from the workload. The goal of the `Online` approach is to obtain as good results as the `Offline`, with the key factor of avoiding the need for profiling. In other words, `Online` should be as efficient as other state-of-the-art approaches, in addition to provide an alternative for the previous profiling experiments bottleneck.

4.1 Model's definitions and architecture

Problem definition: To allocate the appropriate number of VMs in the server cluster and define the configuration of the physical resources to reduce the cluster power consumption without compromising the QoS of the application.

In our model, the main feature is the simplification of the VM allocation process in the cluster, which ensures the agility and scalability needed for scale-out management systems. Besides that, our approach requires only a few simple measurements (utilization rates from the VMs and cores). Therefore, our model does not require specific information about the processor architecture, since the needed measurements are available in most virtualization tools. The proposed model relies on the following three definitions:

Definition 1: Each VM allocated in the cluster is configured as a single default virtual server called Quantum Virtual Machine, or simply QVM;

Definition 2: All QVMs have equal processing capacity, regardless of the core configuration where each one is allocated to run;

Definition 3: Each application is serviced by a Logical Web Server (LWS), which consists of one or more QVMs distributed along the cluster. This amount of QVMs will be set dynamically according to the workload demands to the LWS, and will be limited by the processing capacity of the resources used.

To obtain a homogeneous processing capacity of all active QVMs (**Definition 2**), we must consider the heterogeneous characteristics of the cluster processing resources. The differences between power consumption and performance in heterogeneous processing environments are highlighted in Chapter 3. However, even in homogeneous environments, i.e. cores and servers with the same configurations, the difference in a VM processing capacity is due to several factors, such as the initial VM configuration, number of co-allocated VMs, load balancing between the VMs, etc. Therefore, the normalization of the VMs performance should be viewed in a broader perspective, not only related to homogeneous or heterogeneous processing environments.

As **Definition 3** indicates that a LWS consists of a variable number of QVMs, and **Definition 2** indicates that the set of active QVMs has homogeneous performance, we can conclude that the only factor that will determine the processing capacity of a LWS is the number of QVMs assigned to it. Therefore, the LWSs have an elastic processing capacity according to the workload demand. This elastic behavior enables energy savings by reducing the cluster *idle power*. As seen in Fig. 4.1, LWSs with reduced processing capacity have a small set of active QVMs, and LWSs with high processing capacity have a great amount of active QVMs.

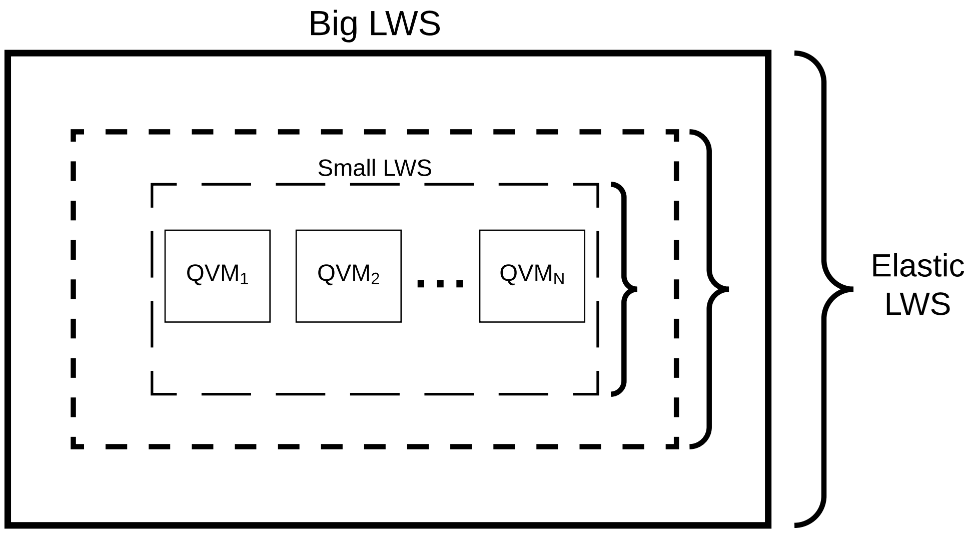


Figura 4.1: Logical Web Server with elastic processing capacity

Our model performs the cluster management via three modules: **Balancer**, **Quantum VM Manager** and **Controller**, as shown in Fig. 4.2. The **Balancer** distributes requests among active QVMs, using a round-robin policy since all active QVMs have the same processing capacity. All procedures associated with managing running QVMs are implemented in the **Quantum VM Manager**.

As other virtualizers, Xen has a native management kit, called Xen Manager Tool. We adapted this tool to work with the QVM agile clone process by extending the Xen API. Finally, the **Controller** module is responsible for monitoring the utilization rate of

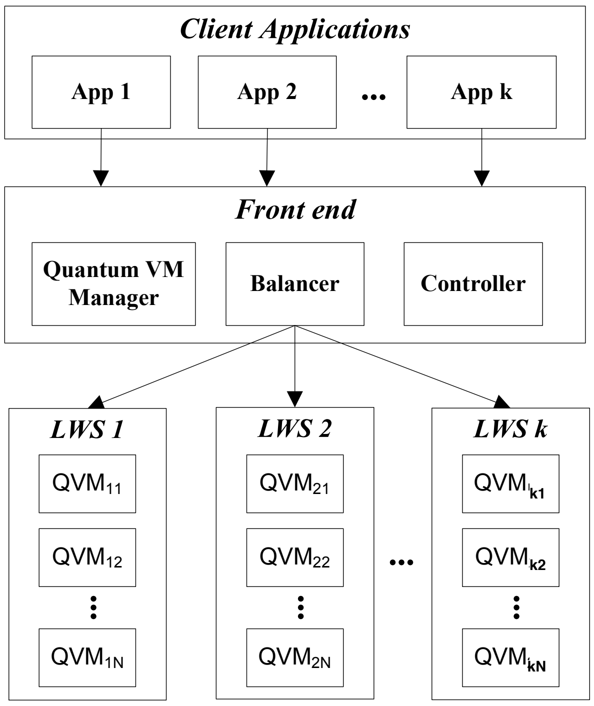


Figura 4.2: The model architecture

the active QVMs and cores. The **Controller** is also responsible for executing the resource managers described in Sections 4.2 and 4.3, which means that it is up to the **Controller** to dynamically define and carry out the reconfiguration actions in the cluster. Since the model's rules and its architecture are described, the resource managers are presented in the next sections.

4.2 The Offline approach

We highlight that this approach was called `Offline` because of its resources profiling requirements, which are done before the operational stage. However, the resource management actions in the `Offline` are performed at runtime, in accordance with its own policies. According to the relationships analyzed in Chapter 3, the normalization of the VMs performance ensured by **Definition 2** is performed in the `Offline` approach by checking the following three related variables. It should be noted that the indexes i and j are associated with QVMs and cores, respectively, since the core is the processing element on the cluster and its operating frequency can be set at runtime. Thus, for instance, QVM$_{ij}$ means the QVM i allocated on the core j in the cluster.

1) UV_{ij} - QVM VCPU utilization rate: This variable points to a direct linear relationship with the processing capacity of the QVM. Although ratified in our `Performance` `vs. CPU Utilization` experiments (Fig. 3.5-a of Chapter 3), this is an intuitive relationship because the greater the number of requests processed, the higher the utilization rate of the QVM.

2) $\frac{f_j}{f_{j_{max}}}$ - Normalized core operating frequency: As the value of $f_{j_{max}}$ on each core j is constant, this variable is impacted only by the current operating frequency f_j of the core. Due to the direct linear relationship between this variable and the QVM processing capacity, if a higher operating frequency is set through the DVFS, the processing capacity of the QVM will be improved, for instance.

3) N_j - Number of co-allocated QVM in a core: This variable has an inverse linear relationship with the processing capacity of the co-allocated QVM, as shown in the `Co-allocation performance evaluation` tests presented in Fig. 3.4 of Chapter 3. This occurs because the greater the amount of co-allocated QVM, the greater the need for sharing the core's resources, which leads to a performance degradation of the running applications.

If we perform a separate analysis for each of the three variables, the result would be an increase or decrease of the processing capacity of a sub-set of QVMs and **Definition 2** would be violated. Therefore, these three variables must be managed to ensure the normalization of the QVMs processing capacity. Furthermore, **Definition 3** ensures the elastic capacity of a LWS is performed only by the increment or decrement of its set of active QVMs, simplifying the QVM allocation process. The processing capacity of the QVM i allocated on the core j is called c_{ij}, and is modeled by Eq. 4.1. In addition to the three mentioned variables, we also define R_j, which is described in details in the end of this section, as a constant associated with the core j's maximum capacity for serving requests.

$$c_{ij} = \frac{UV_{ij}.R_j.\frac{f_j}{f_{j_{max}}}}{N_j} \tag{4.1}$$

The model for power consumption is based on the core operating states: *busy* and *idle*, considering f is the current operating frequency, where for each core $f \in \{f_{min}..f_{max}\}$. As analyzed in the initial tests, we use the linear relationship between power and core utilization rate (U), and the quadratic relationship between power and operating frequency of the core. Thus, the power consumption is modeled according to Eq. 4.2.

$$P(U) = P(idle, f) + (P(busy, f) - P(idle, f)).U \tag{4.2}$$

The values of $P(idle, f)$ and $P(busy, f)$ for all values of f available in the DVFS are obtained by the following equations:

$$P(busy, f) = P(busy, f_{min}) + \Phi_{busy}.(f - f_{min})^2$$

$$P(idle, f) = P(idle, f_{min}) + \Phi_{idle}.(f - f_{min})^2$$

where:
$\Phi_{busy} = P(busy, f_{max}) - \frac{P(busy, f_{min})}{(f_{max} - f_{min})^2}$
$\Phi_{idle} = P(idle, f_{max}) - \frac{P(idle, f_{min})}{(f_{max} - f_{min})^2}$.

In other words, the power of a core depends only on its operating frequency f, its utilization rate U, and the constants $P(busy, f_{min})$, $P(idle, f_{min})$, $P(busy, f_{max})$ and $P(idle, f_{max})$.

The power consumed by a physical server is represented by the sum of the power of its cores. However, there is a part that represents the power consumption of other hardware devices besides the cores, such as power supply, memory, motherboard, disk, etc. This part, which is indicated by the variable Hw, varies according to the workload characteristics, taking into account how each type of workload uses the hardware devices. Thus, the power consumed by a physical server with N cores is $P_{server} = \sum_{k=1}^{N} P_k(U) + Hw$.

The main goal of our power modeling is to predict the power consumption of a core based on only four measured power states described earlier in Eq. 4.2: $P(busy, f_{min})$, $P(idle, f_{min})$, $P(busy, f_{max})$ and $P(idle, f_{max})$, which should be obtained in the profiling stage. Although not eliminating entirely profiling, this method avoids the need for measuring the core power consumption in all its operating frequencies available through DVFS. For all experiments in the profiling stage, we allocated one exclusive QVM on the core where the processing capacity and power consumption are measured. Consequently, the power measurements of each core were performed the same way as in the initial tests (Chapter 3) using the WattsUP Pro power meter.

To observe the accuracy of our analytical power model, which is responsible for providing the estimated power consumption of each processing resource, we compared the power consumption values measured in the profiling stage with the value of $P(U)$ given by Eq. 4.2. The normalized root mean square error for the predicted power was always less than 1.9%. Although the normalized root mean square error is quite small, it should also be mentioned that the maximum absolute error was always less than 9.4%, which validates our analytical power model with reasonable good accuracy. Thus, based on the four power measurements performed in the profiling stage, `Offline` is able to manage each core in the cluster according its power characteristics, and has potential to achieve significant energy savings.

However, the trade-off between power and performance is the crucial issue to be analyzed, in order to provide an energy-efficient processing environment. Therefore, since the power profile of each core is properly defined, we also need to obtain the performance profile of each core. In other words, we need to define the processing capacity of each core available in the cluster. To this end, the constant R_j (Eq. 4.1) is used as the core j's maximum processing capacity in terms of serving requests. To set the value of R_j, we analyzed the saturation point of the QVM performance, observing the rate of requests at which a significant increase in response time occurs. As the QVM is isolated in the core, we can assume that R_j is requivalent to the core j maximum processing capacity.

As mentioned earlier, the key factor of the resource modeling performed by `Offline` are the power analytical model and the processing capacity evaluation, which have to be obtained in a prior experiments stage. In the follow section, we present the `Online` resource manager, which eliminates this a priori profiling in order to define its resource modeling.

4.3 The Online approach

Efforts to develop an accurate power and performance modeling of VMs range from utilization-based methods [70] [64], to approaches based on internal procedures related to the processor architecture to model the power consumption of the VMs [36] and [59]. These approaches, as well as `Offline`, require a previous profiling of the resources to obtain specific informations about the processing platform. As discussed earlier in Chapter 2, can be impractical to stop the operation of a processing environment in order to perform experiments. More to the point, to set the running applications to a hold status is complex, due to the requirement to implement restoring procedures and to handle incomplete requests. Therefore, a runtime approach is needed, which must be dynamically adapted for different types of workloads to model VMs performance and power consumption.

`Online` performs the evaluation of the processing capacity of a core considering its capacity to host co-allocated QVMs. In other words, the more a core can host QVMs with a low variation of the core's utilization rate, the higher its processing capacity is. Furthermore, as all QVMs have equal processing capacity through the normalization process ensured by the **Definition 2** of our model, we can consider that each QVM represents a constant workload. Thus, the number of co-allocated QVMs is a basic metric in our model to analyze the processing capacity of heterogeneous cores, since we define a standard element (QVM) to support the running applications on the cluster.

`Online` is based on control periods in order to set a dynamic weight W_j related to the processing capacity of each core. Accordingly, the variables related to the cluster's resources modeling are presented as follows:

- U_j: Measured value of utilization rate of the core j.

- $\frac{f_j}{f_{j_{max}}}$: Normalized operating frequency of the core j.

- $U_{j_{norm}}$: Normalized utilization rate of the core j based on its measured value of utilization rate U_j and its normalized operating frequency.

- N_j: Number of co-allocated QVMs in the core j.

First, Eq. 4.3 adjusts U_j according to f_j/f_{max} to obtain the value of $U_{j_{norm}}$. This normalization value is required in order to use in our performance modeling an absolute utilization rate, which should be independent of the current operating frequency f_j of the core.

$$U_{j_{norm}} = U_j . \frac{f_j}{f_{j_{max}}} \tag{4.3}$$

Then, a weight W_j is calculated for each active core in the cluster in every control period. This weight is related to the number of co-allocated QVMs on the core j (N_j) and their impact on the normalized utilization rate of the core, as can be seen in Eq. 4.4. The more a core can co-allocate QVMs with a small variation of its normalized utilization rate, the higher its weight is. Thus, as mentioned earlier, W_j has a direct relationship with the processing capacity of each core in the cluster. Finally, as the W_j value is defined, we can model the processing capacity of the QVM i co-allocated on the core j. We used the utilization rate of the QVM's VCPU (UV_{ij}) combined with the weight W_j calculated to the core j. Accordingly, the Eq. 4.5 illustrates the `Online` performance modeling of a QVM.

$$W_j = \frac{N_j}{U_{j_{norm}}} \tag{4.4}$$

$$c_{ij} = UV_{ij}.W_j \tag{4.5}$$

The weight W_j was also used in our power modeling, which combines the value of W_j with the *big/small cores* paradigm. The goal is to properly estimate the energy-efficiency of a core based on its value of W_j. Since the `Online` performance modeling relates W_j with the processing capacity of the core j, we assumed that cores with low values of W_j are associated with *small cores*, which have high energy-efficiency and low processing capacity. Otherwise, cores with high performance (high W_j) were defined as *big cores*, and they present a higher power consumption and performance in comparison with the *small cores*, due to their more complex internal components (larger cache, advanced pipelines, more auxiliary registers, etc.)

The key insight behind this assumption is that robust and high performance processors remain idle most of the time. Thus, there is an obvious waste when the high processing capacity is not utilized to its fullest. Therefore, to keep the active cores with high utilization rates, we arranged the cores available in the cluster according their weights W_j (Eq. 4.4), prioritizing the use of *small cores* to handle the workload demand in the cluster.

As our power and performance modeling of the cluster's resources is based on the weight W_j of each core, a resource manager could properly assign the workload to a set of cores in a dynamic manner, using approaches aiming energy savings or high performance, for instance. Thus, a resource manager designed for energy savings should prioritize the *small cores* (low values of W_j) to assign the current workload, since they are more energy-efficient than *big cores*. Otherwise, the resource manager should activate the cores with high values of W_j (*big cores*) in order to obtain high performance. Even so, a mix of *small cores* and *big cores* could be used to observe the trade-off between power consumption and performance. These points are detailed and analyzed in the next sections.

Since the resource modeling of each resource manager have been described, as well as their relationship with the three model's **Definitions**, we can summarize the `Offline` and `Online` modeling approaches as follows:

	Performance modeling	**Power modeling**
`Offline`	Constant R_j previously measured for each core in the cluster and associated to its processing capacity	Analytical power modeling based on four previous power measurements for each core in the cluster
`Online`	Each core j performance is evaluated as the capacity to host co-allocated QVMs, defined by the weight W_j	Cores arranged by their energy-efficiency, considering their weight W_j

Tabela 4.1: Resource modeling of the two implemented resource manager

In the next sections, we present how each approach implements its resource management policies, based on the modeling of the processing resources. Thus, we describe how each core and server is managed in order to implement the cluster reconfiguration actions.

4.4 Resource management policies

As the scope of our work is limited to the power management of processing resources in the cluster, our model does not consider other power consumption factors such as network equipments, cooling system, etc. Thus, the power consumption of the cluster can be modeled as the power consumption of its physical servers. For a cluster with S servers, the problem of minimizing the cluster power consumption can be modeled as: $min \sum_{j=1}^{S} P_{server_j}$. To achieve this goal, we deactivate the cores and servers that are not hosting QVMs, using the hibernation techniques of the processor combined with low power procedures, as described in [42] and [65], and the server on/off operation, respectively.

To prevent the *idle power*, we set the threshold of 30% as a lower bound for the core utilization rate. So, if a core achieves a utilization rate less than 30% it will be deactivated. However, the deactivation is performed only if the remaining set of active cores can support the reallocation of the QVMs that were co-allocated on the deactivated core. A trigger to activate a new core is also required, since the current number of active QVMs co-allocated in the same core can degrade their own performance, as analyzed in Chapter 3. To this end, we determined the threshold of 80% which is related to the saturation point of the core observed in the initial tests. Therefore, when the utilization rate of a core is higher than 80%, the resource manager actives a new core in the cluster. Other threshold bounds were analyzed in our preliminary experiments, but the chosen rates of 30% and 80% were more suitable for the trade off between power and performance.

In the `Offline` approach, we define the sequence for core activation/deactivation using the values of power consumption which were measured in the previous profiling stage. Accordingly, the most energy-efficient core will be the one with the highest ratio value between the number of attended requests and power consumed. Therefore, all the cores in the cluster are ordered by their energy-efficiency in a static list, since the energy-efficiency metric has a constant value for each core. The goal is to first activate the more energy-efficient cores.

It should be noticed that the sequential activation of cores using their energy-efficiency can lead to an unexpected higher power consumption scenario. This occurs because there may be an inactive core, or a set of inactive cores, where the set or a sub-set of QVMs could be consolidated to achieve lower power consumption, even though the energy-efficient sequential activation has been performed. This is analyzed by calculating the power consumption in the current configuration using Eq. 4.2, and by verifying if any inactive core can consolidate the QVMs workload or a part of it, using Eq. 4.1, with lower power consumption. Then, if the consolidation is feasible, we clone the appropriate amount of QVMs in the destination core and deactivate the source cores.

As mentioned earlier, in the `Online` approach we prioritize the activation of the *small cores* rather than the *big cores*. To this end, we analyzed the variable W_j to arrange the activation/deactivation sequence of cores. As the weight W_j assigned for each core by the `Online` is related to its processing capacity, a low W_j represents a *small core*, while a great W_j represents a *big core*. It is important to notice that unlike the static list used by the `Offline`, the sequence for activation/deactivation of cores in the `Online` approach is dynamic, since the W_j values assigned to each active core are adjusted at every control period. Finally, as the sequence for activation/deactivation of processing resources in both approaches is analyzed, the next section presents the QVM allocation algorithm used for the reconfiguration actions in the cluster.

4.5 The QVM allocation problem

The QVM allocation problem can be formulated as: **When and where should the proper amount of QVMs that perform a LWS should be allocated, in order to maximize the energy savings without violating the QoS agreements?**

As discussed in Chapter 2, this is a classic optimization problem, which attempts to find a cluster configuration able to minimize power consumption restricted to performance requirements. However, as analyzed in details in the next chapter (Performance Evaluation), the resource managers based on optimization need a massive computation effort to find a good solution. This computational effort is usually lengthy, which does not enable a fine-grained QoS control for the running applications in the cluster. Thus, a simpler but also effective solution for the allocation problem is required. This simplification is achieve through the QVM paradigm, and the reconfiguration actions performed by our resource managers are described as follows.

In order to predict rapid increases of request loads and to determine a new cluster configuration that prevents QoS violations, we have to verify the greatest latency among the reconfiguration actions defined in our resource managers. The worst case will determine how often (in seconds) a reconfiguration action must be done so that the cluster can be properly sized for the expected load. In our case, for both approaches, the greatest latency is the server on/off operation, followed by the activation of a new core and the cloning of a set of QVMs.

The on/off server operation is based on ACPI *wake on LAN*, and has latency lower than 1 second. Thus, we specified in the reconfiguration policy a 1-second control period. This value is not too small to execute many configuration actions in a short period of time, causing instability in the applications QoS, and is not large enough to require a more complex and accurate forecasting mechanism.

The work described in [66] shows that the amount of requests sent to a server can be mapped by linear models. The linear regression used in the cited work achieves good accuracy levels even when the prediction is performed for extended periods in the future. Therefore, the k^{th} control period in our prediction process uses a linear regression based on λ previous control periods to predict the amount of requests sent to the cluster in the period $k{+}1$. We used a linear regression with $\lambda = 10$ which analyzes the number of requests received by the cluster at the front-end layer. This λ value was defined by preliminary evaluation tests, and was sufficient to properly predict the peak loads occurring during the performance evaluation of our model (Chapter 5).

To achieve the consolidation of QVMs, we used the ordered lists of cores assembled by the `Offline` and `Online` approaches, which were implemented in a vector $V[1..N]$, as mentioned in the previous section. We assume that there are M active cores in the set of N available cores in the cluster. Then, a sequential resources activation is performed by the QVM allocation algorithm, which uses the average utilization rate of the LWSs (set of active QVMs assigned to each LWS), called UV_{avg} to determine the need for cloning/destruction of QVMs.

Using the average utilization rate can mitigate any distortions of the processing load on each QVM, due to the dynamic nature of the requests and the normalization of the QVMs performance. Therefore, maximum and minimum utilization thresholds of the set of QVMs, called UV_{max} and UV_{min}, were designated with values of 70% and 80%, respectively, in order to keep high utilization rates. As analyzed through the initial experiments present in Chapter 3, the use of high utilization rates is more energy-efficient than setting high operating frequencies through DVFS to increase the application performance.

The algorithm starts verifying if QVMs should be cloned or destroyed dynamically to maintain the restriction of the LWSs utilization rate ($UV_{min} \leq UV_{avg} \leq UV_{max}$). Then, a sub-set of QVMs will be cloned if $UV_{avg} > UV_{max}$ to improve the LWSs processing capacity for avoiding violation of QoS. Similarly, if $UV_{avg} < UV_{min}$, a subset of QVMs must be destroyed to provide energy savings.

To set the amount of QVMs that must be cloned or destroyed, we define $C(k)$ as the processing capacity of the set of QVMs at k^{th} period. We denominate $\Delta C = C(k+1) - C(k)$ as the processing capacity of the set of QVMs to be cloned or destroyed at the $(k+1)^{th}$ period. Therefore, $\Delta C < 0$ indicates how many QVMs must be destroyed, $\Delta C > 0$ how many must be cloned, and $\Delta C = 0$ represents that the set of active QVMs should not be changed to $(k+1)^{th}$ control period. Then, we verify if the set of active cores can fully support the current $(C(k))$ and the predicted (ΔC) workload for the next control period, based on the utilization rate of the set of active cores.

If none new core has to be activated, the clone QVMs (related to ΔC) are allocated on the set of active cores ordered in $V[1..M]$. If a new core is needed, we allocate the clone QVMs related to ΔC on the next available core in $V[1..N]$. Finally, if a LWS is oversized, we deallocate the set of QVMs related to ΔC from the core $V[M]$, and this core is deactivated if it no longer hosts QVMs. Lastly, the workload consolidation procedure is performed for the `Offline` approach, and a 1-second control loop pause is performed to finish the current control period.

The basic steps of the QVM allocation algorithm are described as following:

Step 1: Verify if $UV_{avg} > UV_{max}$ or $UV_{avg} < UV_{min}$.

Step 2: Calculate ΔC.

Step 3: Check if the set of active cores can support ΔC, and redefine the set of active cores if the condition is false.

Step 4: Clone or destroy a sub-set of QVMs based on the value of ΔC.

In order to provide more details about the QVM allocation algorithm, its pseudo-code is presented in Fig. 4.3.

```
1: Reconfiguration Algorithm ($V[N]$)
2: loop
3:    Get $UV_{avg}$ from the set of QVMs
4:    Calculate $\Delta C$
5:    if $UV_{avg} > UV_{max}$ then
6:       if $\sum_{j=1}^{M} R_j \geq (\Delta C + C(k))$ then
7:          Allocate $\Delta C$ on $V[1..M]$
8:       else
9:          Activate core $V[M+1]$
10:         Allocate $\Delta C$ on $V[M+1]$
11:         $M \leftarrow M + 1$
12:      end if
13:   else if $UV_{avg} < UV_{min}$ then
14:      Deallocate $\Delta C$ from $V[M]$
15:      if $V[M] = \emptyset$ then
16:         Deactivate core $V[M]$
17:         $M \leftarrow M - 1$
18:      end if
19:   end if
20:   Consolidation procedure ($V[N]$)
21:   sleep (1second) {control loop pause}
22: end loop
```

Figura 4.3: Reconfiguration algorithm

Basicly, we must verify how many QVM should be cloned or destroyed dynamically to maintain valid the $UV_{min} \leq UV_{avg} \leq UV_{max}$ restriction. Thus, a sub-set of QVM is cloned if $UV_{avg} > UV_{max}$ (line 5) to improve the LWS processing capacity for avoiding violation of QoS. Similarly, if $UV_{avg} < UV_{min}$, a subset of QVM must be destroyed (line 13) for providing energy savings. To set the amout of QVM that must be cloned or destroyed, we define $C(k)$ as the processing capacity of the QVM set in k^{th} period. Then, if we denominate as ΔC the processing capacity of the set of QVM to be cloned or deactivated in the $(k+1)^{th}$ period, we have: $\Delta C = C(k+1) - C(k)$.

As our proposed model, the resource management policies and the reconfiguration actions are presented, we describe in the next chapter a performance evaluation through real tests in our testbed.

Chapter 5

Performance Evaluation

Our model was compared with `Performance`, `PowerSave` and `OnDemand` CPU governors from Linux, with state-of-the-art models based on optimization, which manage the VM allocation using the *bin packing* problem, and a `PMC` approach. In the Linux governors all servers and cores remain active, `Performance` always sets the cores to their maximum operating frequencies. Otherwise, `Powersave` sets the cores to their minimum operating frequencies and `OnDemand` adjusts the operating frequencies of the cores according to the workload, but does not make use of server on/off.

When the `Performance` governor is used, the cluster operates on its maximum processing capacity, since all the cores are set with their highest operating frequencies available on DVFS. Moreover, is expected that `Performance` achieves the best results for applications performance, but the worst for energy savings in comparison with the other approaches. On the other hand, `PowerSave` is designed for minimizing the cluster power consumption, setting the cores to operate on their lowest operating frequecies. However, the impact on the applications performance should be much higher than other approaches.

The dynamic management of the trade-off between power and performance is the goal of `OnDemand`. To this end, `OnDemand` sets at runtime the operating frequency of a core based on its utilization rate. More to the point, `OnDemand` provides a fair comparison with our model, since it implements a dynamic management of power and performance based on the current workload.

Although the Linux governors used in our experiments enable relevant energy savings for the processing environment, we can found several works in the literature with significantly better results. Therefore, in order to evaluate the real efficiency of our model, it should be compared with state-of-the art approaches, which are presented as follows.

5.1 State-of-the-art approaches implemented

As analyzed in Chapter 2, a very common approach observed in power and performance management in virtualized server clusters regards determining the most suitable configuration of the active VMs by means of an optimization process. These works define the VM allocation process as a *bin packing* problem. The objective is to minimize the cluster power consumption, without violating the application's QoS agreements. Thus, in the next section we describe the optimization approaches implemented in our performance evaluation.

5.1.1 Optimization approaches

`Optimization` approaches should define the optimal number of active cores and servers, and set the operating frequencies based on the tuple *(frequency, utilization)* of each core. `Optimization` uses the same power consumption modeling as `Offline`, to define and set the proper operating frequency of a core. Thus, `Optimization` estimates the cluster power consumption for several combinations of cluster configuration, based on the power measurements performed in the profiling stage. Then, it sets the cluster configuration that is able to: (*i*) handle the workload demand without compromising the QoS; (*ii*) with the lowest power consumption.

The objective function is modeled to minimize the cluster power consumption. The constraint equations must observe the maximum processing capacity of a server s, which is the sum of the processing capacity of its N cores (1), and the limitations imposed by the QoS of the applications (2), which is related to the utilization rate of the set of M active cores. Since each VM is assigned to an active core, there is no co-allocation of VMs in the same core and the VM processing capacity is the same as the core. Therefore, the utilization rate of a core (U_j) is equal to the utilization rate of its hosted VM. To implement its resource management `Optimization` uses server on/off, DVFS, core activation/deactivation and *live migration* to perform the reallocation of VMs in the cluster.

$$minimize \sum_{k=1}^{M} \sum_{j=1}^{N} P_j(U) = P_j(idle, f_j) + (P_j(busy, f_j) - P_j(idle, f_j)).U_j \qquad \forall f_j \in \{f_{min}..f_{max}\} \tag{5.1}$$

Subject to:

$C_s \leq \sum_{j=1}^{N} R_j$ (1)

$U_j \leq 0.8, \forall j \in \{1..M\}$ (2)

It should be noted that `Optimization` also needs a previous profiling of the cluster's resources. As can be seen in Eq. 5.1 and in restriction (1), the values of R_j, $P_j(idle, f)$ and $(P_j(busy, f)$ are required to perform the `Optimization` process. Thus, the mentioned approach uses the same values as `Offline` does to model and manage the cluster.

We used the Gurobi Optimizer 5.6 [26] to solve the optimization problem, and analyzed two `Optimization` schemes. `Default-Opt` uses the default parameters configuration of the Gurobi, setting the amount of time spent in heuristics to 5%. Thus, `Default-Opt` will spend 5% of its runtime on heuristics. The second scheme, called `Heuristic-Opt`, is more aggressive in the use of heuristics to find the best solution for the optimization problem. The goal is to reduce the time spent to define the best cluster configuration, since the heuristics can achieve a good solution for the optimization problem and decrease the required computational effort, as discussed in Section 2. To this end, in `Heuristic-Opt` we set the time spent in heuristics to 20%, which is good rate to keep the balance between the heuristics and the solving processing, as described in [9] and [57].

In next section, we present a Performance Monitoring Counters (PMC) approach to estimate power consumption and performance of a core via analytic models, which avoid the need for previous profiling, as discussed as follows.

5.1.2 PMC-based approach

Performance counters on chip are generally accurate, and they provide significant insight into the processor performance at the clock-cycle granularity. Besides that, PMC are already incorporated into and exposed to user space on most modern processors architectures. The PMC capability for accurately estimating at runtime power consumption and performance enables the resource manager to make better real-time decisions. This is the main reason for its wide implementation in resource modeling solutions described in the literature.

`PMC` approach performs the resource modeling using a set of performance counters that are related with the performance and power consumption of the core. As the state-of-the-art approaches based on monitoring counters, `PMC` performs the power consumption modeling of the core j as: $P_{relative} = (\sum_{i=1}^{i=comps} AR_i.P_i) + P_{static}$, where P_i is the weight of component i that we need to solve, and AR_i is its activity ratio. The $AR_i.P_i$ represents the dynamic power consumption of the component i, and P_{static} represents the overall static power consumption of all components.

A set of performance counters is available in the modern processors platforms. We selected the counters that represent relevant information about performance and power consumption modeling. As described in [68], the dynamic power consumption of modern processors depends on the circuit switching activity in their cores and the NorthBridge (NB). This cited work points out that main contributors to the power consumption of multi-core processors are the processing units (FPU, ALU, etc.) and the off-chip cache. Moreover, the power modeling attribute part of the NB's power to each core, since they share it. Thus, based on the state-of-the-art PMC approaches, the four performance counters listed below are able to capture enough information to indicate the relative power consumption of a core: *(i)* **L2 Cache Miss**; *(ii)* **Retired UOPS**, *(iii)* **Retired MMX and FP instructions**, and *(iv)* **Dispatch stalls**.

The **L2 cache miss** rate indicates how often the NB (NorthBridge) is used, since the last level cache (L3) is an off-core device and is accessed always when a L2 cache miss occurs. Then, the **Dispatch stalls** can be also used to approximate the NB activity caused by a core. We attribute part of the NB power to each core, since they share it. To do this, we can only choose NB-related PMC that can be collected on a per-core basis, rather than any events counted in the shared NB. As **L2 Cache Miss** represents L3 cache access operations from the core measuring the events, the NB usage is direct related.

The **Dispatch stalls** is usually caused by load/store queues or OoO (Out of Order) storage (e.g., reservation stations) being full. Load/store queue stalls are usually due to the long latency of the last-level cache or off-chip memory accesses, which happen in the NB. Therefore, as **L2 Cache Miss**, we also found that **Dispatch stalls** can help approximate the NB activity caused by a core.

The **Retired UOP** and **Retired MMX** means that, in CPU pipeline, the instruction is finally executed, and its rate is related to the CPU utilization. In other words, these two components show the utilization of the microcircuits in the core chip. Finally, it is necessary to assign the weight P_i to each one of the four selected components. As the PMC are related to a particular processor architecture, we choose the weights associated to the AMD architecture described in [68], since our testbed have servers with this processor type and the weights were obtained through real tests performed by the cited work. All the four counters have direct relationship with power, except for the retired FP/MMX instructions PMC. This is expected, since such instructions have higher latencies and this class of instructions reduces the throughput of the system, resulting in lower power use.

Since the power modeling through performance counters is defined, it is necessary to estimate the processor performance. As shown in [77], the CPI (Cycles Per Instructions) correlates well with the application behavior. CPI is a function of the hardware platform, and it is a reasonably stable measure over time. Moreover, CPI presents a direct relationship with the changes in compute-intensive application behavior, and it is more accurate than similar PMC, such as IPS (instructions Per Second) and cache misses, for instance.

The resource management policies used by `PMC` are similar to `Online` (described in Chapter 4). Thus, we define the same utilization bounds of 30% and 80% to indicate the need for deactivating a core for energy savings, and to activate a new core due to overload, respectively. In order to provide a fair comparison between `PMC`, `Optimization` and `QVM` approaches, the PMC-based resource manager also uses server on/off, DVFS, core activation/deactivation and *live migration* to perform the reallocation of VMs in the cluster. Moreover, `PMC` also prioritizes the activation of the *small cores* rather than the *big cores*, using its own resource modeling to define the *small* and *big cores* sets.

As the `Optimization` and `PMC` models have been described, our testbed and implementation details are presented as follows.

5.2 The testbed

The infrastructure of our testbed is composed of: *(i)* one server to run the HTTP request generator module; *(ii)* one server to host the front-end layer, and *(iii)* three servers to implement the virtualized application servers cluster. Table 5.1 describes the configuration of these servers. As the scalability of our work is a key factor in the performance evaluation, the application servers have a significant number of cores to host large sets of QVMs. Thus, as explained below, the three servers in the application cluster have together a total of 176 cores, enabling the scalability analysis with a suitable setting. All servers use CentOS Linux 6 as the operating system, and have the Xen 3.1 version installed to support the QVMs. The remote manipulation of the QVMs from the **Quantum VM Manager** module was implemented with XML/RPC, since Xen API also uses these calls in its internal procedures. The **Balancer** is based on the *mod-proxy-balance* module of *Apache*.

Server	**Role**	**Processor**	**No. of cores**	**Memory**
Oahu	Load generator	Intel Core i7	2	8 GB
Kauai	Front-end	AMD Bulldozer	8	8 GB
Magnus 1	Application server	4 x AMD Opteron 1,7 GHz	48	32 GB
Magnus 2	Application	4 x AMD Opteron 1,8 GHz	64	32 GB
Magnus 3	Application	4 x AMD Opteron 2,1 GHz	64	32 GB

Tabela 5.1: Configuration of the machines in our testbed

To ratify that our model is independent from the processing platform, we used two different test scenarios. The first scenario consists of a `Big/Small Cores` platform and does not use the DVFS to dynamically change the operating frequency of the cores. As our testbed does not have real *small cores*, we simulate them by setting different operating frequencies in sub-set of cores: 29 cores with 0.6 GHz, 29 cores with 0.8 GHz and 30 cores with 1.0 GHz. We then have 88 cores (half of the testbed) in the set of *small cores*, and the other half in the set of *big cores*. For the set of big cores, we used the nominal operating frequencies shown in Table 5.1. The second test scenario consists of a set of heterogeneous cores with different processing capacities and power consumption, in which the DVFS is used by all the evaluated approaches, except for `Performance` and `PowerSave`, which always sets the highest and lowest operating frequencies of the cores, respectively.

To provide a better analysis of the results, first we present the experiments using only web applications, since they are the major focus of scale-out data centers and cloud computing platforms. Then, in order to analyze the efficiency of the approaches in different scenarios, all types of application workloads available on the CloudSuite 2.0 are evaluated.

5.3 Web applications scenario

Web applications is the main service in the cloud. Traditional web services with dynamic and static content are moved into cloud computing platforms to provide fault-tolerance and scalability. Although many variants of the traditional web applications are used in these environments, the underlying architecture remains unchanged. Clients requests are handled by stateless web servers which either directly serve static files from disk or perform the processing of dynamic content. To perform the tests in the web applications scenario we used the **Web Serving** workload from the CloudSuite benchmark. The clients send requests to login to a social network, and also perform different types of requests, such as profile update, new posts, search by content, among others. All these requests must be handled by the web servers layer, where the resource managers presented in the earlier sections perform their policies for the cluster management.

In order to perform the clients' action, accessing the web applications, we developed a HTTP request generator module that will trigger requests to the web servers. To simulate the execution of three simultaneous web applications in the cluster, we used different access traces to Google applications available in [23], as shown in Fig 5.1. We then have three LWSs implemented in the cluster for our performance evaluation, each one supporting a web application. Intervals of 100 minutes were selected in the files analyzed. The original traces were adjusted to the processing capacity of our testbed, which is measured in requests per second, normalizing the original values to the interval ranging from 0 to 180 simultaneous requests per application.

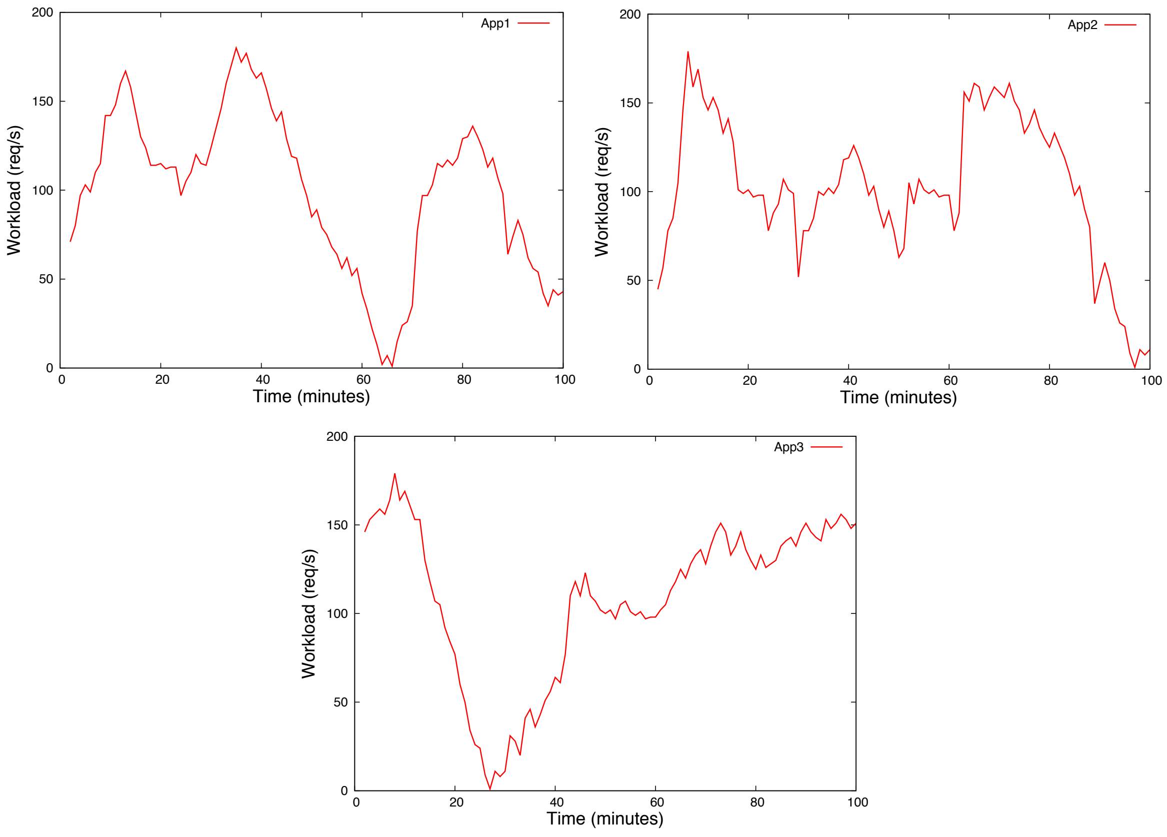
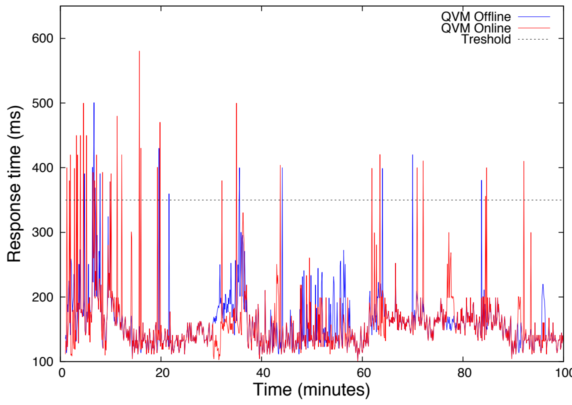
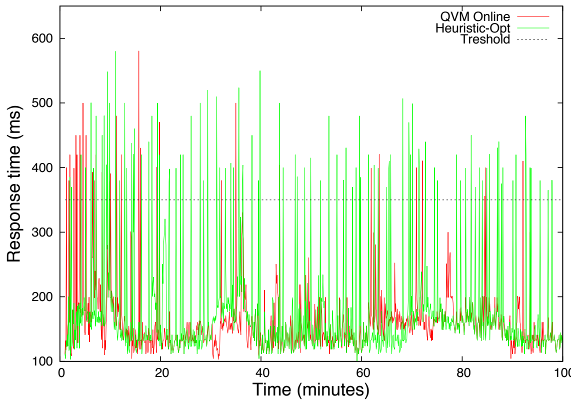


Figura 5.1: Application workloads

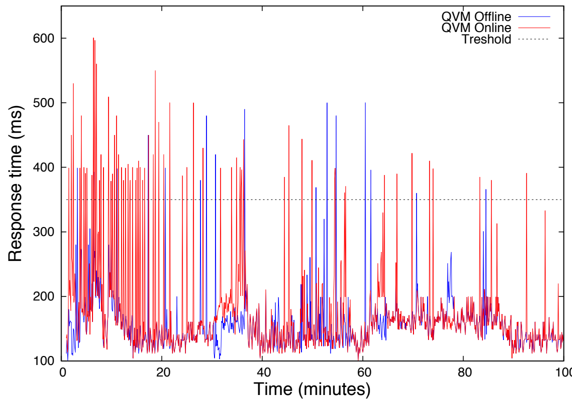
As a metric to evaluate the results of each approach we used: *(i)* power consumed by the virtualized web servers cluster and *(ii)* QoS violation rate of the applications (*QoS-violation*). In addition to these two metrics, specifically for the `Optimization` approaches and for our model, the time spent to set the reconfiguration actions is also compared in order to evaluate their scalability. The CloudSuite `Web Serving` benchmark has a default value of QoS for web requests, which is equivalent to 350 ms in the tests performed. Thus, the *QoS-violation* corresponds to the sum of all requests that exceeded the response time



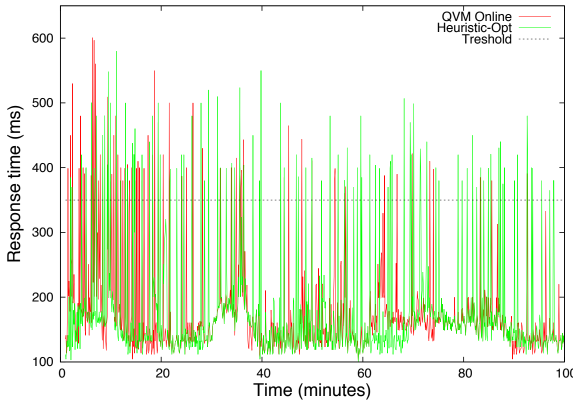
(a) Big/Small Cores



(b) Big/Small Cores



(c) Heterogeneous Cores



(d) Heterogeneous Cores

Figura 5.2: Application performance

threshold, divided by the number of requests served during the tests.

Fig. 5.2 depicts the applications' response time, in both platform `Big/Small Cores` and `Heterogeneous Cores`. To better present the data on each performance graph, a comparison between two approaches in both platforms is provided. The first comparison is between the `Offline` and `Online` resource managers proposed in our model, as shown in Fig. 5.2-a and 5.2-c. As observed, `Online` has a higher rate of *QoS-violation* than `Offline`. The reason for this behavior is that `Online` has no information about the cluster's resources, as does `Offline` when its previous profiling is performed. As `Online` performs at runtime its resource modeling, even for the cores which the performance modeling was already executed, it needs to recalculate the weights assigned for each core in every control period.

Therefore, the *QoS-violation* observed in `Online` at the beginning of the tests occurs due the lack of information about the resources available in the processing platform, mainly on the `Heterogeneous Cores` where the runtime modeling is more complex due to the DVFS operations. However, about twenty minutes after starting the test, the

QoS-violation rate of `Online` achieved a similar behavior as `Offline`, which means that the runtime resource modeling is very accurate about the real processing capacity of the cluster.

The second comparison is presented in Fig. 5.2-b and 5.2-d, and is between `Online` and `Heuristic-Opt`, since the results of `Heuristic-Opt` are better than `Default-Opt` for the *QoS-violation*. For the `Big/Small Cores` platform the *QoS-violation* of `Heuristic-Opt` is 7.3%, while the rate of `Online` is 1.0%. When we analyze the `Heterogeneous Cores` platform, the *QoS-violation* rate of `Heuristic-Opt` is 9.9%, and `Online` achieves a rate of 1.4%. Therefore, the *QoS-violation* of `Heuristic-Opt` is much higher than `Online`. This occurs due to the long time (9.3 seconds on average) spent to define the best cluster configuration by `Heuristic-Opt`, which makes the fine-grained QoS control impracticable.

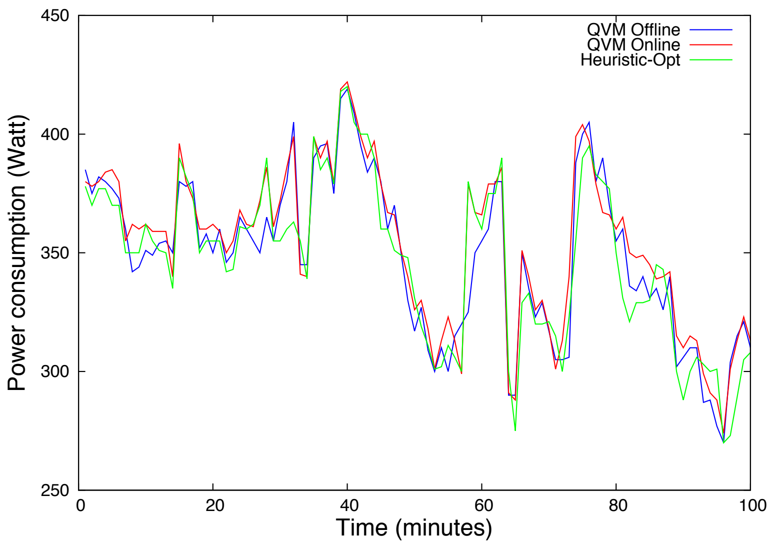
To analyze in details how agile our resource managers are in comparison with the approaches based on `Optimization`, Table 5.2 summarizes the average time spent to define the cluster configuration in both platforms. As observed, the average time spent by `Optimization` approaches is much longer than others resource managers. These results ratify that `Optimization` approaches are not able to set a fine-grained QoS control, which leads to the high *QoS-violation* rate observed along the tests performed. It should be noticed that even the more aggressive heuristics used in `Heuristic-Opt` are not able to significantly reduce the time spent to set the cluster configuration. Therefore, this behavior is more related to the combinatorial nature of the optimization problem than to the use of procedures to simplify the solving processing.

No. of VMs	**Default-Opt**	**Heuristic-Opt**	**PMC**	**QVM Offline**	**QVM Online**
10	33 ms	33 ms	0.017 ms	0.016 ms	0.016 ms
20	331 ms	280 ms	0.017 ms	0.016 ms	0.016 ms
40	3328 ms	1933 ms	0.018 ms	0.016 ms	0.017 ms
100	155239 ms	62228 ms	0.018 ms	0.017 ms	0.017 ms

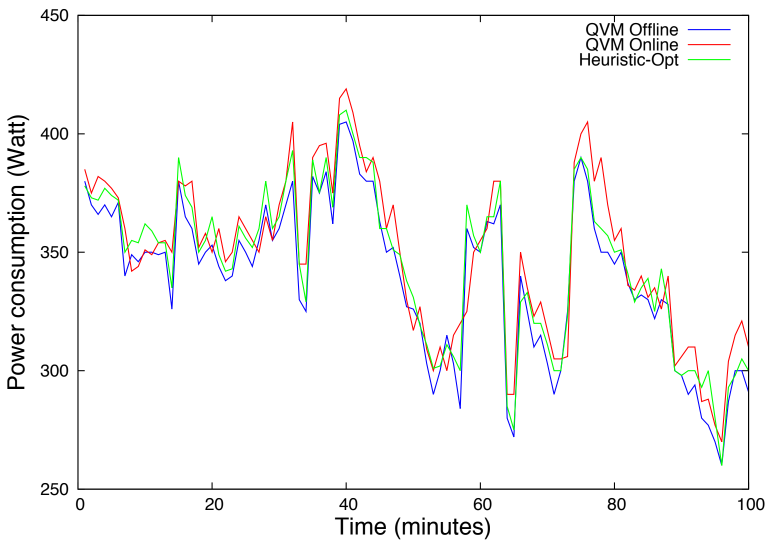
Tabela 5.2: Average time spent to define the cluster configuration

Resource manager	`Heterogeneous Cores`	`Big/Small Cores`
`Performance`	0%	–
`PowerSave`	16.9%	–
`OnDemand`	22.5%	–
`Default-Opt`	46.7%	48.1%
`Heuristic-Opt`	48.9%	49.8%
`PMC`	41.5%	42.6%
`QVM Offline`	52.4%	54.3%
`QVM Online`	51.1%	52.2%

Tabela 5.3: Energy savings results



(a) Big/Small Cores



(b) Heterogeneous Cores

Figura 5.3: Web server cluster power consumption

Fig. 5.3 shows the web servers cluster power consumption when `Offline`, `Online` and `Heuristic-Opt` are used. For a better visualization, we provide the average measurement values in each 1-minute interval, and the values of the Linux governors and `Default-Opt` were omitted. However, the results of all approaches for energy savings and *QoS-violation* are summarized in Table 5.3 and Table 5.4, respectively. The tests were executed five times in order to calculate the mean and standard deviation of the values measured. The values shown in both tables have a confidence interval of 90%, calculated using a Student's t distribution with 4 degrees of freedom.

Since `Performance` is related with the maximum processing capacity of the cluster, it was used as baseline in comparison with the results achieved with other approaches.

Resource manager	`Heterogeneous Cores`	`Big/Small Cores`
`Performance`	0%	–
`PowerSave`	19.3%	–
`OnDemand`	1.9%	–
`Default-Opt`	12.8%	10.5%
`Heuristic-Opt`	9.9%	7.3%
`PMC`	6.5%	6.1%
`QVM Offline`	1.2%	0.7%
`QVM Online`	1.4%	1.0%

Tabela 5.4: QoS-violation results

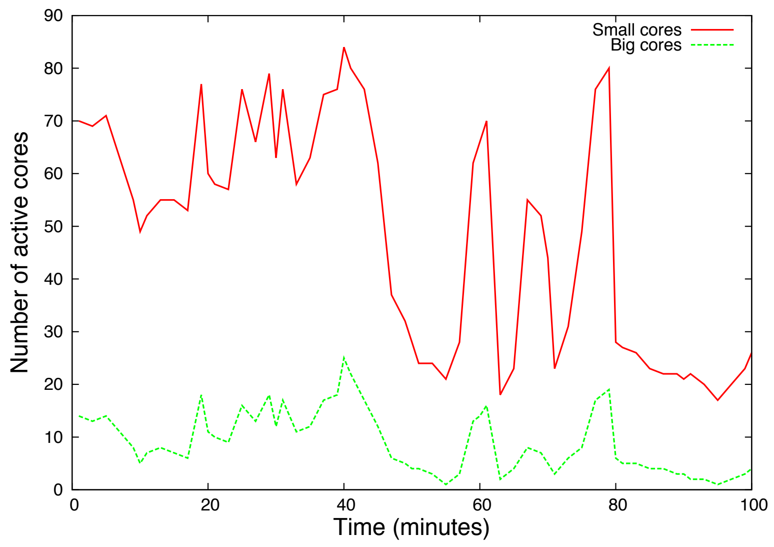


Figura 5.4: Number of big/small cores

As can be seen in Table 5.3, our resource managers achieved more energy savings than `Optimization` approaches. Moreover, the trade-off between power and performance managed by each approach should be observed. While `Optimization` approaches exhibit high rates of *QoS-violation*, `Offline` and `Online` resource managers are able to keep the *QoS-violation* in a very low level regardless the processing platform analyzed.

The Linux CPU governors were evaluated only in `Heterogeneous Cores` platform, since they are DVFS-based approaches and the `Big/Small Cores` environment does not enable DVFS operations. Besides that, it is interesting to notice the results of `PowerSave` and `OnDemand` for energy savings. As `PowerSave` sets the lowest operating frequency for all cores in the cluster, the turnaround time of the requests are much higher than observed on `OnDemand`, for instance, which adjusts the operating frequency of each core according the current workload. This issue was discussed in Chapter 3, where we empirically analyzed the trade-off between power and performance using DVFS in our initial experiments. Therefore, as observed in these experiments, despite the lower instant power consumption achieved by `PowerSave`, the average power consumed along the time is significantly higher, since the cores must remain longer activated for processing the workload.

Finally, as can be seen in Tables 5.3 and 5.4, the results for energy savings and *QoS-violation* are slightly better in `Big/Small Cores` than in `Heterogeneous Cores`. This indicates that the flexible management provided through DVFS operations is not more efficient than cores with constant operating frequencies arranged in different sets according their characteristics. Thus, grouping the cores into different sets, one composed by cores with low power consumption and other of high performance elements, as the *big/small cores* paradigm does, enables an efficient resource management aimed to energy savings.

In order to verify if `Online` in fact prioritizes the use of *small cores* in its resources management policy, we analyzed the number of *small cores* and *big cores* that are activated throughout the performance evaluation using the `Big/Small Cores` platform. This analysis is shown in Fig. 5.4, which depicts the average number of *big* and *small cores* used during the test period. As can be seen, the number of active *small cores* is much higher than the number of *big cores.*

Even during the peak workload scenarios, at about 40 and 80 minutes, the number of *small cores* is higher than *big cores.* Although some *big cores* are still used in low workload scenarios, due to some margin of error of `Online` modeling, we ratify the assumption presented in Chapter 3, which shows that *small cores* are more energy-efficient than *big cores* for web applications. In other words, even without any previous information about the cluster's resources, `Online` achieves similar results as `Offline` through the prioritization of the *small cores* in the allocation of QVMs on the cluster.

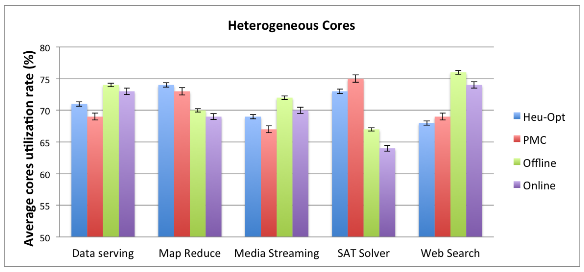
5.4 Other CloudSuite workloads

The five other CloudSuite 2.0 workloads consist of applications that have been selected based on their popularity in large-scale data centers. These benchmarks are based on real-world processing scenarios and represent popular online services, as listed below:

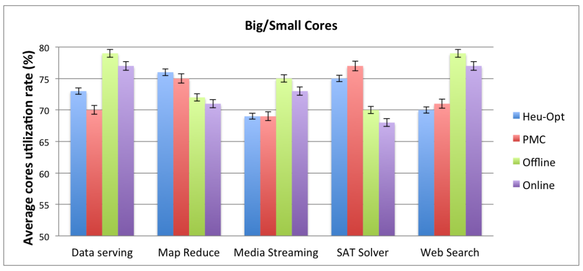
- **I/O bound workloads:** This group includes **Data Serving**, **Media Streaming** and **Web Search**. These workloads represent massive data maniuplation with tight latency constraints, such as today's most popular video-sharing website features, for instance.
- **CPU bound workloads:** The main characteristic of **Map Reduce** and **Sat**

Solver workloads are the large-scale computation-intensive tasks. Thus, these workloads are focused on CPU usage, with irrelevant I/O procedures.

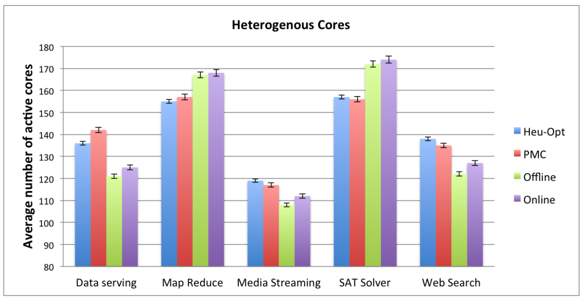
To analyze the energy-efficiency of the evaluated approaches in these scenarios, we show how the *idle power* impacts on the cluster power consumption along the test period. To this end, Fig. 5.5-a and 5.5-b show the average utilization rate of the active cores in the cluster. Then, Fig. 5.5-c and 5.5-d present the number of active cores during the test, and Fig. 5.5-e and 5.5-f show the average cluster power consumption. As in web application scenario, in all power measurements we used the WattsUP Pro [75] to collect the power consumption of the whole server cluster. To provide a better visualization, the graphics show the results for the 4 best-performing approaches (`Online`, `Offline`, `PMC` and `Heu-Opt`).



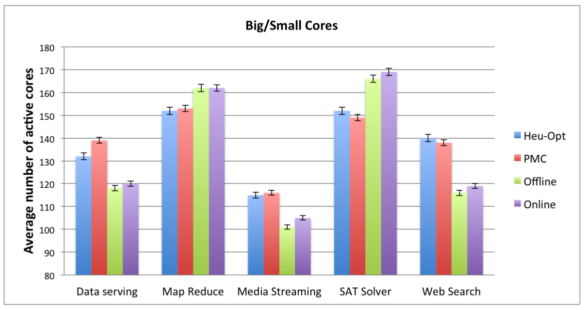
(a) Heterogeneous Cores



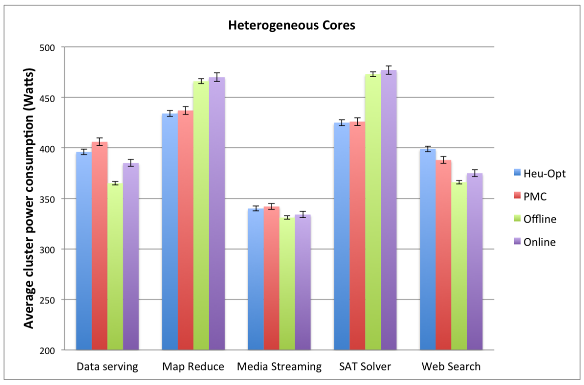
(b) Big/Small Cores



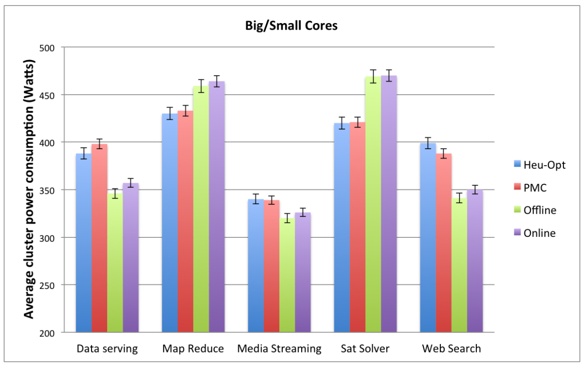
(c) Heterogeneous Cores



(d) Big/Small Cores



(e) Heterogeneous Cores

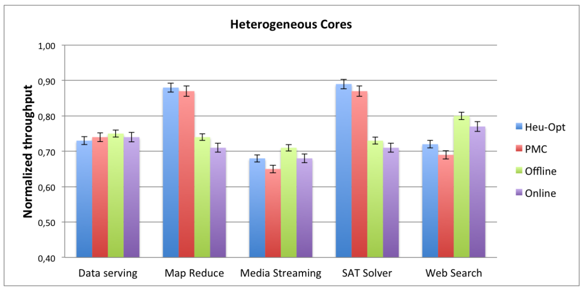


(f) Big/Small Cores

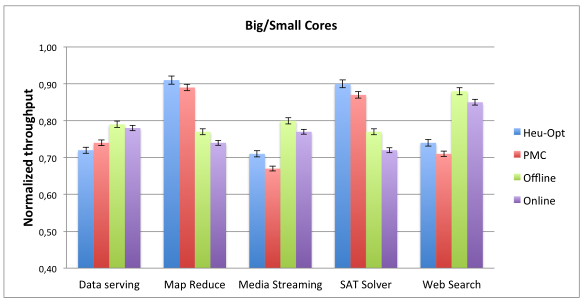
Figura 5.5: Power measurements

As can be seen in Fig. 5.5-a and 5.5-b, for I/O bound workloads (**Data Serving**, **Media Streaming** and **Web Search**) `Offline` and `Online` enable a higher occupation of the cores available in the cluster than other approaches. Therefore, a fewer number of active cores is required by both resource managers to handle the workload (Fig. 5.5-c and 5.5-d), leading to a lower power consumption (Fig. 5.5-e and 5.5-f). Moreover, the results of `Offline` and `Online` are even better on the `Big/Small Cores` platform when the I/O bound workloads are performed. Thus, as observed in the **Web Serving** scenario described in the early section, these results ratify the energy-efficiency of `Big/Small Cores` platform.

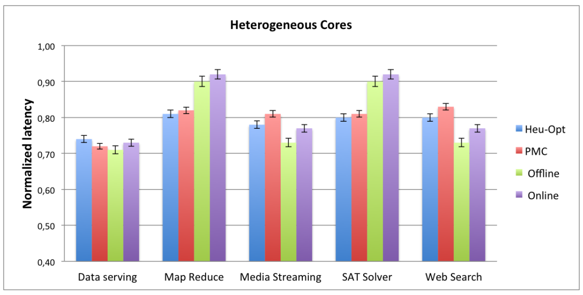
On the other hand, when **Map Reduce** and **SAT Solver** workloads are performed, which have a CPU bound behavior, `Heuristic-Opt` and `PMC` provide better results for energy savings than approaches based on QVM. This occurs due to the power and performance modeling used by both approaches. As `PMC` models the processing capacity through processor counters, and `Optimization` approaches are mainly based on the core processing capacity, these methods seem to be more accurate in *CPU bound* workloads.



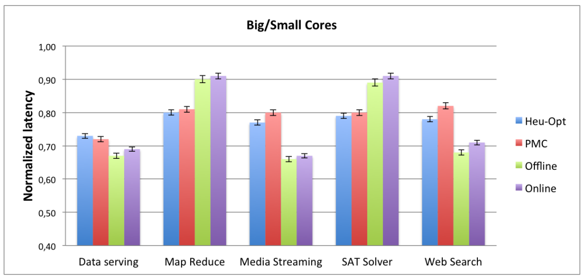
(a) Heterogeneous Cores



(b) Big/Small Cores



(c) Heterogeneous Cores



(d) Big/Small Cores

Figura 5.6: Performance measurements

Fig. 5.6 shows how the applications performance is impacted by each resource management. We analyzed the normalized throughput (Fig. 5.6-a and 5.6-b) and the normalized latency of the requests attended by the application servers cluster (Fig. 5.6-c and 5.6-d) for each workload. Both metrics use the average values observed along the test period. The throughput is defined by the number of requests handled per second, and the latency indicates the time needed to complete the request. We chose these two metrics rather than the *QoS-violation* used in **Web Serving** workload in order to provide a more embracing performance evaluation, since the *QoS-violation* is a metric strictly related to web applications.

We used `Performance` and `PowerSave` Linux governors as baseline for the *throughput* and *latency* metrics, respectively. Thus, the maximum throughput is obtained when `Performance` is used, which is associated with the value 1 in the normalized throughput. As `PowerSave` always sets the operating frequency of a core to its minimum value, in order to maximize the energy savings, it obtains the maximum latency for serving requests in comparison with other approaches. We associate the latency of `PowerSave` with the value 1 in the normalized latency.

As observed, the results of `Offline` and `Online` are similar for the applications performance. As for the power measurements, the better results of both approaches for the applications performance occur when I/O bound workloads are performed. Another common feature with the power experiments is the result of `PMC` and `Heuristic-Optimization` for the applications performance during the CPU bound workloads. As can be seen in Fig. 5.6, when **Map Reduce** and **SAT Solver** workloads are performed, `PMC` and `Heuristic-Optimization` show similar results. Moreover, these results are significantly better than QVM approaches.

Resource manager	`Heterogeneous Cores`	`Big/Small Cores`
`Performance`	0%	–
`PowerSave`	15.7%	–
`OnDemand`	21.7%	–
`Default-Opt`	45.8%	46.9%
`Heuristic-Opt`	47.7%	48.2%
`PMC`	40.1%	41.3%
`QVM Offline`	52.7%	53.9%
`QVM Online`	51.62%	52.1%

Tabela 5.5: Energy savings results

The results of all approaches are summarized in Table 5.5 and Table 5.6, where the average values obtained using all the workloads (**Web Serving** excluded) for energy savings and applications performance are presented, respectively. As occurred in the **Web Serving** workload, here we can see that the `Big/Small Cores` platform provides more energy savings than `Heterogeneous Cores`. Additionally, despite the metric of applications performance has changed compared to the **Web Serving** scenario (from *QoS-violation* to *Normalized Throughput* and *Normalized Latency*, the `Big/Small Cores`

Resource manager	**Normalized Throughput (HC)**	**Normalized Throughput (B/S C)**	**Normalized Latency (HC)**	**Normalized Latency (B/S C)**
`Performance`	1.0	–	0.0	–
`PowerSave`	0.0	–	1.0	–
`OnDemand`	0.60	–	0.83	–
`Default-Opt`	0.71	0.72	0.73	0.72
`Heuristic-Opt`	0.73	0.75	0.73	0.71
`PMC`	0.72	0.74	0.74	0.72
`QVM Offline`	0.74	0.75	0.74	0.72
`QVM Online`	0.73	0.74	0.74	0.73

Tabela 5.6: Performance in Heterogeneous Cores (HC) and Big/Small Cores (B/S C)

also surpasses `Heterogeneous Cores` from a performance perspective.

	`QVM Online`	Petrucci et al. [58]	`Bertran et al.` [10]	`Souza et al.` [67]	`Kusic et al.` [39]
Approach	VM normalization	Optimization	PMC	Heuristics	Optimization
VM migration	Agile clone	Live migration	Live migration	No virtualization	Live migration
Need for previous profiling	No	Yes	No	No	Yes
Fine-grained QoS control	Yes	No	Yes	Yes	No
Scalability	Linear	Exponential	Linear	Linear	Exponential
Energy savings	52.3%	52.1%	49.8%	30.3%	26.0%
QoS-violation	1.2%	9.72%	6.7%	2.5%	1.6%

Tabela 5.7: Comparison with state-of-the-art approaches

Finally, to observe how our model is situated on the energy-efficient clusters management area, we provide a holistic analysis considering the key features addressed by the most relevant related works. To this end, Table 5.7 presents a comparison between the `Online` and some state-of-the-art approaches designed for power and performance management in application servers clusters, which were discussed in our Related Work chapter. The results of the mentioned works were obtained directly from the original papers, since it was not viable to implement those approaches precisely in our testbed.

The energy savings in `Petrucci et al.` [58] and `Bertran et al.` [10] are similar to that obtained with `Online`. However, our approach is able to simultaneously achieve other relevant goals, such as avoiding previous profiling of the cluster's resources, linear scalability and performing a fine-grained applications QoS control. Moreover, `Online` provides the lowest QoS violations in comparison with the analyzed works. Therefore, in addition to achieving good results for energy savings, our proposed model offers a more comprehensive solution for the management of virtualized application server clusters.

Chapter 6

Conclusions

The model proposed in this thesis showed good results for power management, achieving energy savings up to 51.6% over a cluster designed for a peak workload scenario. Our model and the `Optimization` schemes had a similar behavior with respect to power consumption. However, our resource managers perform the energy savings with negligible impact on application performance, achieving a fine-grained QoS control by relying on a few simple measurements (utilization rates from the VMs and cores).

Additionally, as the experiments were performed in two different platforms, the assumption that our model is independent of the processing environment was ratified. Furthermore, our `Online` approach achieves good results performing a dynamic modeling of the cluster's resources, avoiding the need for previous profiling and prioritizing the use of the *small cores* instead of the *big cores*.

Another important factor regarding the model was its scalability, especially when the time spent to define the reconfiguration actions is compared with the `Optimization` approaches. As mentioned earlier, the *bin packing* problem used as the basis of the optimization process evaluated has an exponential scale, in contrast to our model which performs the management of the cluster's resources based on an algorithm with linear complexity. Thus, in large data centers scenarios and cloud computing platforms with thousands of machines and several applications, for example, the proposed model is feasible to perform a global management of power and performance.

As future work we intend to analyze the fault-tolerant issues, in order to implement in our model some restoring procedures. Thus, a checkpoint control can be used to define different stages of processing data, which is crucial for long processing tasks such as specific jobs, and scientific computing, for instance. However, checkpoint procedures impact on application performance, due to the need to persist data for future recovery procedures, as analyzed in [69], [71] and [16]. Thus, the period to perform the checkpoint must be detailed and analyzed to avoid QoS degradation. This is a relevant requirement in scale-out data centers and cloud computing environments, which must be capable to properly manage the processing environment when energy blackouts or servers crash occur, for instance.

Another relevant issue for future work is to provide a specialized configuration of the QVMs according the workload. Once the `PMC` and `Optimization` presented better results for CPU bound workloads, some improvements could be done in the QVM paradigm for CPU intensive applications. Since CPU intensive applications perform long processing windows, the QVMs could be consolidate into larger VMs. However even these larger VMs should be based on the normalization of the VMs performance implemented by our model. This approach could avoid the great number of active VMs, reducing the need for scheduling the VMs on the cores available in the server cluster. Thus, the processing windows could be longer, avoinding the context switch among the VMs, which is a crucial issue in CPU bound workloads.

As described in the Introduction, we summarize as follows how each key research issue was addressed by our model. In addition, we also discuss the goals achieved by this thesis.

6.1 Questions and answers about key issues of the proposed model

1: Why the proposed model is independent from the processing platform? Although our reconfiguration algorithm considers the processor core as the processing resource, it does not mean a dependency on a multicore architecture. The core is related to our model as a logical element, and it can be mapped to any physical processing resource, such as single-core processores, for instance. To this end, it is enough to indicate in the implementation phase where each logical core is physical located (server A, processor B, i.e.), in order to enable its activation/deactivation and other managing procedures. Moreover, our model does not require measurements specifically related to the processor's configuration, such as IPC, cache misses, among others. As our proposed approach is based only on measurements related to VMs and cores occupation, it could be easily implemented in most processing platforms.

2: Does DVFS have a relevant role for energy savings in our model? The answer is no. The tests performed in the `Big/Small Cores` platform show the efficiency of our model regardless of wether the DVFS technique was applied or not. Additionally, the results for energy savings and applications performance were better using the `Big/Small cores` platform, then when the DVFS was enabled in the `Heterogeneous Cores` scenario. However, our model does not become unfeasible with DVFS-based processing platforms. As can be seen in Chapter 5 (Performance Evaluation), the proposed model is suitable for both platforms and does not require specific adjustments to operate on each of them.

3: Why the model is scalable?
Our model is based on algorithm with linear complexity for performing the cluster reconfiguration actions. Therefore, even for a great number of active VMs to be evaluated and managed, the fine-grained QoS control was performed through a 1-second control loop. This issue was explicitly shown by the time spent to define the cluster configuration for each resource management approach (Table 5.2 - Chapter 5). As our model presents an average time of 0.016 ms for any analyzed number of active VMs, it is able to operate the virtualized cluster in an agile manner. This is ratified by the rate of *QoS-violation* achieved by our model, which is significantly lower than other approaches. More to the point, the testbed used in our performance evaluation experiments is composed of 176 cores, which enabled a proper scalability analysis.

4: Why the model is agile to perform reconfiguration actions?
The first point is the use of VM agile clone for creation or migration of VMs in the cluster. As described in our initial tests in Chapter 3, the VM agile clone is able to provide a significant reduction on the latency of the VM allocation process, in comparison with the VM live migration procedure. Other relevant issue is the simple monitoring procedures, which need only to collect the utilization rates of cores and VMs. Thus, we do not need to use complex operations, and most of time also intrusive, to collect information about the processing environment, such as performance monitoring counters (PMC), for instance. With a simple and lightweight monitoring, there is more room to implement a shorter control period, which facilitates the energy savings and the fulfilment of SLA agreements.

5: How the QVM paradigm enables the simplification of the cluster management?
The three **Definitions** which our model is based on, specially the **Definition 2**, provides the VM performance normalization. Thus, the proposed model works considering homogeneous processing capacity for all active QVMs, and the only decision to take in the resource management is if we should increase, decrease, or keep stable the number of

active QVMs for each LWS. In other words, all the available reconfiguration actions, such as DVFS, core activation/deactivation, agile clone, etc. are used as basis for providing the QVM performance normalization abstraction. Therefore, this simplification with regard to performance modeling permeates other elements present in the model. For example, the `Balancer` uses a *round-robin* policy to distribute the requests among the cluster, since all the processing elements (QVMs) have the same processing capacity, avoiding complex policies for load balancing.

Referências

[1] Amazon. Amazon ec2. http://aws.amazon.com/ec2, December 2013.

[2] Amur, H.; Nathuji, R.; Ghosh, M.; Schwan, K.; Lee, H.-H. S. Idlepower: Application-aware management of processor idle states. In *Proceedings of the Workshop on Managed Many-Core Systems, MMCS* (2008. DOI: http://dx.doi.org/10.1109/ICCAD.2011.6105394), vol. 8, Citeseer.

[3] Barham, P.; Dragovic, B.; Fraser, K.; Hand, S.; Harris, T.; Ho, A.; Neugebauer, R.; Pratt, I.; Warfield, A. Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review 37*, 5 (2003), 164–177. DOI: http://dx.doi.org/10.1145/1165389.945462.

[4] Barroso, L. A.; Hölzle, U. The case for energy-proportional computing. *IEEE Computer 40*, 12 (2007), 33–37. DOI: http://doi.ieeecomputersociety.org/10.1109/MC.2007.443.

[5] Belady, C. In the data center, power and cooling costs more than the it equipment it supports. http://www.electronics-cooling.com/articles/2007/feb/a3/, February 2007.

[6] Beloglazov, A.; Abawajy, J.; Buyya, R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems 28*, 5 (2012), 755–768. DOI: http://dx.doi.org/10.1016/j.future.2011.04.017.

[7] 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), IEEE Computer Society, pp. 826–831.

[8] Berl, A.; De Meer, H. An energy consumption model for virtualized office environments. *Future Generation Computer Systems 27*, 8 (2011), 1047–1055.

[9] Berthold, T. Measuring the impact of primal heuristics. *Operations Research Letters 41*, 6 (2013), 611–614. DOI: http://dx.doi.org/10.1016/j.orl.2013.08.007.

[10] Bertran, R.; Becerra, Y.; Carrera, D.; Beltran, V.; Gonzàlez, M.; Martorell, X.; Navarro, N.; Torres, J.; Ayguadé, E. Energy accounting for shared virtualized environments under dvfs using pmc-based power models. *Future Generation Computer Systems 28*, 2 (2012), 457–468.

[11] Bircher, W. L.; John, L. K. Complete system power estimation using processor performance events. *Computers, IEEE Transactions on 61*, 4 (2012), 563–577.

[12] Borkar, S. Thousand core chips: a technology perspective. In *Proceedings of the 44th annual Design Automation Conference* (2007), ACM, pp. 746–749. DOI: http://dx.doi.org/10.1145/1278480.1278667.

[13] Chakraborty, K.; Roy, S. Topologically homogeneous power-performance heterogeneous multicore systems. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011* (2011), IEEE, pp. 1–6.

[14] CloudSuite. Cloud suite benchmark. http://parsa.epfl.ch/cloudsuite/cloudsuite.html, August 2013.

[15] Dhiman, G.; Mihic, K.; Rosing, T. A system for online power prediction in virtualized environments using gaussian mixture models. In *Design Automation Conference (DAC), 2010 47th ACM/IEEE* (2010), IEEE, pp. 807–812.

[16] Duolikun, D.; Aikebaier, A.; Enokido, T.; Takizawa, M. Power consumption models for migrating processes in a server cluster. In *Network-Based Information Systems (NBiS), 2014 17th International Conference on* (2014), IEEE, pp. 15–22.

[17] Esmaeilzadeh, H.; Blem, E.; Amant, R. S.; Sankaralingam, K.; Burger, D. Power challenges may end the multicore era. *Communications of the ACM 56*, 2 (2013), 93–102. DOI: http://dx.doi.org/10.1145/2408776.2408797.

[18] Fan, X.; Weber, W.-D.; Barroso, L. A. Power provisioning for a warehouse-sized computer. In *ACM SIGARCH Computer Architecture News* (2007), vol. 35, ACM, pp. 13–23.

[19] Ferdman, M.; Adileh, A.; Kocberber, O.; Volos, S.; Alisafaee, M.; Jevdjic, D.; Kaynak, C.; Popescu, A. D.; Ailamaki, A.; Falsafi, B. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. *ACM SIGARCH Computer Architecture News 40*, 1 (2012), 37–48.

[20] Gandhi, A.; Harchol-Balter, M.; Das, R.; Lefurgy, C. Optimal power allocation in server farms. In *ACM SIGMETRICS Performance Evaluation Review* (2009), vol. 37, ACM, pp. 157–168.

[21] Ghribi, C.; Hadji, M.; Zeghlache, D. Energy efficient vm scheduling for cloud data centers: Exact allocation and migration algorithms. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on* (2013), IEEE, pp. 671–678.

[22] Gmach, D.; Rolia, J.; Cherkasova, L.; Kemper, A. Resource pool management: Reactive versus proactive or let's be friends. *Computer Networks 53*, 17 (2009), 2905–2922.

[23] Google. Google apps. www.google.com/apps, September 2013.

[24] green grid consortium, T. Green grid. www.greengrid.org, october 2014.

[25] Gupta, V.; Schwan, K. Brawny vs. wimpy: Evaluation and analysis of modern workloads on heterogeneous processors. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International* (2013), IEEE, pp. 74–83. DOI: http://dx.doi.org/10.1109/IPDPSW.2013.130.

[26] Gurobi. Gurobi optimizer 5.6. http://www.gurobi.com, Junho 2014.

[27] Hager, G.; Treibig, J.; Habich, J.; Wellein, G. Exploring performance and power properties of modern multi-core chips via simple machine models. *Concurrency and Computation: Practice and Experience 28*, 2 (2014), 189–210. DOI: http://dx.doi.org/10.1002/cpe.3180.

[28] Hines, M. R.; Gopalan, K. Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments* (2009), ACM, pp. 51–60.

[29] Hirofuchi, T.; Nakada, H.; Itoh, S.; Sekiguchi, S. Enabling instantaneous relocation of virtual machines with a lightweight vmm extension. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on* (2010), IEEE, pp. 73–83.

[30] Hölzle, U. Brawny cores still beat wimpy cores, most of the time. *IEEE Micro 30*, 4 (2010).

[31] Kahng, A. B.; Kang, S.; Kumar, R.; Sartori, J. Enhancing the efficiency of energy-constrained dvfs designs. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 21*, 10 (2013), 1769–1782.

[32] Kansal, A.; Zhao, F.; Liu, J.; Kothari, N.; Bhattacharya, A. A. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM symposium on Cloud computing* (2010), ACM, pp. 39–50.

[33] Keramidas, G.; Spiliopoulos, V.; Kaxiras, S. Interval-based models for run-time dvfs orchestration in superscalar processors. In *Proceedings of the 7th ACM international conference on Computing frontiers* (2010), ACM, pp. 287–296.

[34] Kim, J.; Ruggiero, M.; Atienza, D.; Lederberger, M. Correlation-aware virtual machine allocation for energy-efficient datacenters. In *Proceedings of the Conference on Design, Automation and Test in Europe* (2013), EDA Consortium, pp. 1345–1350.

[35] Kim, K. H.; Beloglazov, A.; Buyya, R. Power-aware provisioning of virtual machines for real-time cloud services. *Concurrency and Computation: Practice and Experience 23*, 13 (2011), 1491–1505.

[36] Kim, N.; Cho, J.; Seo, E. Energy-credit scheduler: an energy-aware virtual machine scheduler for cloud systems. *Future Generation Computer Systems 32* (2014), 128–137.

[37] Kim, W.; Gupta, M. S.; Wei, G.-Y.; Brooks, D. System level analysis of fast, per-core dvfs using on-chip switching regulators. In *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on* (2008), IEEE, pp. 123–134.

[38] Krioukov, A.; Mohan, P.; Alspaugh, S.; Keys, L.; Culler, D.; Katz, R. Napsac: Design and implementation of a power-proportional web cluster. *ACM SIGCOMM Computer Communication Review 41*, 1 (2011), 102–108. DOI: http://dx.doi.org/10.1145/1851290.1851294.

[39] Kusic, D.; Kephart, J. O.; Hanson, J. E.; Kandasamy, N.; Jiang, G. Power and performance management of virtualized computing environments via lookahead control. *Cluster Computing 12*, 1 (2009), 1–15.

[40] Lagar-Cavilla, H. A.; Whitney, J. A.; Scannell, A. M.; Patchin, P.; Rumble, S. M.; De Lara, E.; Brudno, M.; Satyanarayanan, M. Snowflock: rapid virtual machine cloning for cloud computing. In *Proceedings of the 4th ACM European conference on Computer systems* (2009), ACM, pp. 1–12.

[41] Lampka, K., et al. Keep it slow and in time: Online dvfs with hard real-time workloads. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2016), IEEE, pp. 385–390.

[42] Le Sueur, E.; Heiser, G. Slow down or sleep, that is the question. In *USENIX Annual Technical Conference* (2011).

[43] Lee, Y. C.; Zomaya, A. Y. Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing 60*, 2 (2012), 268–280.

[44] Leverich, J.; Monchiero, M.; Talwar, V.; Ranganathan, P.; Kozyrakis, C. Power management of datacenter workloads using per-core power gating. *Computer Architecture Letters 8*, 2 (2009), 48–51.

[45] Liu, H.; Jin, H.; Xu, C.-Z.; Liao, X. Performance and energy modeling for live migration of virtual machines. *Cluster computing 16*, 2 (2013), 249–264.

[46] Lovász, G.; Niedermeier, F.; de Meer, H. Performance tradeoffs of energy-aware virtual machine consolidation. *Cluster Computing 16*, 3 (2013), 481–496.

[47] Ma, D.; Bondade, R. Enabling power-efficient dvfs operations on silicon. *Circuits and Systems Magazine, IEEE 10*, 1 (2010), 14–30.

[48] Ma, K.; Li, X.; Chen, M.; Wang, X. Scalable power control for many-core architectures running multi-threaded applications. In *ACM SIGARCH Computer Architecture News* (2011), vol. 39, ACM, pp. 449–460.

[49] Maciel, P.; Callou, G.; Tavares, E.; Sousa, E.; Silva, B., et al. Estimating reliability importance and total cost of acquisition for data center power infrastructures. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on* (2011), IEEE, pp. 421–426.

[50] Monteiro, A. F.; Azevedo, M. V.; Sztajnberg, A. Virtualized web server cluster self-configuration to optimize resource and power use. *Journal of Systems and Software 86*, 11 (2013), 2779–2796.

[51] Monteiro, A. F.; Loques, O. Qmapper: Scalability and energy saving for virtualized web server clusters. In *Cluster Computing (CLUSTER), 2014 IEEE International Conference on* (2014), IEEE, pp. 270–278.

[52] Monteiro, A. F.; Loques, O. Quantum virtual machine: A scalable model to optimize energy savings and resource management. In *Computer Architecture and High Performance Computing (SBAC-PAD), 2015 27th International Symposium on* (2015), IEEE, pp. 194–201.

[53] Monteiro, A. F.; Loques, O. Scalable model for dynamic configuration and power management in virtualized heterogeneous web clusters. In *Applied Computing (SAC), 2015 ACM Internation Symposium on* (2015), ACM, pp. 146–149.

[54] Monteiro, A. F.; Loques, O. Qmodel: A dynamic approach for power and performance modeling in virtualized servers clusters. In *Applied Computing (SAC), 2017 ACM Internation Symposium on* (2017), ACM, pp. 122–127.

[55] Monteiro, A. F.; Loques, O. Quantum virtual machine: a novel approach for managing power and performance in virtualized web servers clusters. *Journal of Parallel and Distributed Computing under review* (2017).

[56] Nikolaev, R.; Back, G. Perfctr-xen: a framework for performance counter virtualization. In *ACM SIGPLAN Notices* (2011), vol. 46, ACM, pp. 15–26.

[57] Panigrahy, R.; Talwar, K.; Uyeda, L.; Wieder, U. Heuristics for vector bin packing. *http://www.research.microsoft.com* (2011).

[58] Petrucci, V.; Carrera, E. V.; Loques, O.; Leite, J. C.; Mosse, D. Optimized management of power and performance for virtualized heterogeneous server clusters. In *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on* (2011), IEEE, pp. 23–32.

[59] Petrucci, V.; Loques, O.; Mossé, D. Lucky scheduling for energy-efficient heterogeneous multi-core systems. In *Proceedings of the 2012 USENIX conference on Power-Aware Computing and Systems* (2012), USENIX Association, pp. 7–7.

[60] Pettey, C. Gartner estimates ict industry accounts for 2 percent of global co2 emissions. *Dostupno na: https://www. gartner. com/newsroom/id/503867 14* (2007), 2013.

[61] Pricopi, M.; Muthukaruppan, T. S.; Venkataramani, V.; Mitra, T.; Vishin, S. Power-performance modeling on asymmetric multi-cores. In *Compilers, Architecture and Synthesis for Embedded Systems (CASES), 2013 International Conference on* (2013), IEEE, pp. 1–10.

[62] Ranganathan, P.; Leech, P.; Irwin, D.; Chase, J. Ensemble-level power management for dense blade servers. In *ACM SIGARCH Computer Architecture News* (2006), vol. 34, IEEE Computer Society, pp. 66–77.

[63] Rizvandi, N. B.; Taheri, J.; Zomaya, A. Y. Some observations on optimal frequency selection in dvfs-based energy consumption minimization. *Journal of Parallel and Distributed Computing 71*, 8 (2011), 1154–1164.

[64] Rodero, I.; Jaramillo, J.; Quiroz, A.; Parashar, M.; Guim, F. Towards energy-aware autonomic provisioning for virtualized environments. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing* (2010), ACM, pp. 320–323.

[65] Rountree, B.; Ahn, D. H.; de Supinski, B. R.; Lowenthal, D. K.; Schulz, M. Beyond dvfs: A first look at performance under a hardware-enforced power bound. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International* (2012), IEEE, pp. 947–953.

[66] Santana, C.; Leite, J. C.; Mossé, D. Load forecasting applied to soft real-time web clusters. In *Proceedings of the 2010 ACM Symposium on Applied Computing* (2010), ACM, pp. 346–350.

[67] Sousa, L.; Leite, J.; Loques, O. Green data centers: Using hierarchies for scalable energy efficiency in large web clusters. *Information Processing Letters 113*, 14 (2013), 507–515. DOI: http://dx.doi.org/10.1016/j.ipl.2013.04.010.

[68] Su, B.; Gu, J.; Shen, L.; Huang, W.; Greathouse, J. L.; Wang, Z. Ppep: Online performance, power, and energy prediction framework and dvfs space exploration. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture* (2014), IEEE Computer Society, pp. 445–457.

[69] Sztajnberg, A.; Granja, R. S.; Cesário, J.; Monteiro, A. F. A. An integration experience of a software architecture and a monitoring infrastructure to deploy applications with non-functional requirements in computing grids. *Software: Practice and Experience 41*, 1 (2011), 103–127.

[70] Takahashi, S.; Takefusa, A.; Shigeno, M.; Nakada, H.; Kudoh, T.; Yoshise, A. Virtual machine packing algorithms for lower power consumption. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on* (2012), IEEE, pp. 161–168.

[71] Verma, A.; Pedrosa, L.; Korupolu, M.; Oppenheimer, D.; Tune, E.; Wilkes, J. Large-scale cluster management at google with borg. In *Proceedings of the Tenth European Conference on Computer Systems* (2015), ACM, p. 18.

[72] Von Laszewski, G.; Wang, L.; Younge, A. J.; He, X. Power-aware scheduling of virtual machines in dvfs-enabled clusters. In *2009 IEEE International Conference on Cluster Computing and Workshops* (2009), IEEE, pp. 1–10.

[73] Wang, Y.; Wang, X.; Chen, M.; Zhu, X. Power-efficient response time guarantees for virtualized enterprise servers. In *Real-Time Systems Symposium, 2008* (2008), IEEE, pp. 303–312.

[74] Wang, Y.; Wang, X.; Chen, Y. Energy-efficient virtual machine scheduling in performance-asymmetric multi-core architectures. In *Proceedings of the 8th International Conference on Network and Service Management* (2012), International Federation for Information Processing, pp. 288–294.

[75] WattsUp. Watts up meter pro. http://www.wattsupmeters.com, July 2013.

[76] Xie, X.; Jiang, H.; Jin, H.; Cao, W.; Yuan, P.; Yang, L. T. Metis: a profiling toolkit based on the virtualization of hardware performance counters. *Human-centric Computing and Information Sciences 2*, 1 (2012), 8.

[77] Zhang, X.; Tune, E.; Hagmann, R.; Jnagal, R.; Gokhale, V.; Wilkes, J. Cpi 2: Cpu performance isolation for shared compute clusters. In *Proceedings of the 8th ACM European Conference on Computer Systems* (2013), ACM, pp. 379–391.