Virtual Machine Migration Energy Consumption Simulation in Cloud Computing

by

Vincenzo De Maio

submitted to the Faculty of Mathematics, Computer Science and Physics of the University of Innsbruck in partial fulfillment of the requirements for the degree of doctor of science

advisor: Assoz.-Prof. Priv.-Doz. Dr. Radu Prodan,
Institute of Computer Science,
University of Innsbruck.

Certificate of authorship/ originality

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Vincenzo De Maio, Innsbruck on the August 29, 2018

To my family, a safe harbour in the storm.
To all the nice people I met in Innsbruck, whose friendship has been an unconquerable summer in those cold winters.

Contents

Abstract

Energy consumption has become a significant issue for data centres. For this reason, many researchers currently focus on developing energy aware algorithms to improve their energy efficiency. However, due to the difficulty of employing real data centres' infrastructure for assessing the effectiveness of energy-aware algorithms, researchers resort on simulation tools. These tools require precise and detailed models for virtualized data centres in order to deliver accurate results. In recent years, many models have been proposed, but most of them either do not consider energy consumption related to virtual machine (VM) migration or do not consider some of the energy impacting components (e.g. CPU, network, storage). In this work, I focus on increasing the accuracy of existing energy prediction models, by providing the research community with a more accurate data centre energy consumption simulator. To this end, I focus on designing an accurate model of energy consumption of VM migration.

First, I present a comparative analysis of the energy consumption of the software stack of two of today's mostly used network interface cards (NICs) in data centres, Ethernet and Infiniband. I carefully design for this purpose a set of benchmark experiments to assess the impact of different traffic patterns and interface settings on energy consumption. Using these benchmarking results, I derive an energy consumption model for network transfers and evaluate its accuracy for a VM migration scenario. I also propose guidelines for NIC selection from an energy efficiency perspective for different application classes.

Then, I show that omitting VM migration and workload variation from the models could lead to inaccurate consumption estimates. For this rea-

son, I propose a new model for data centre energy consumption that takes into account the previously omitted model parameters and provides accurate energy consumption predictions for paravirtualised VMs running on homogeneous hosts. The new model's accuracy is evaluated with a comprehensive set of operational scenarios. With the use of these scenarios I present a comparative analysis of my model with similar state-of-the-art models for energy consumption of VM migration, showing an improvement up to 24% in accuracy of prediction.

Finally, I propose a new model for data centre energy consumption that takes into account the previously omitted components and provides more accurate energy consumption predictions compared to other state-of-the-art solutions for paravirtualized VMs. I evaluate this model's accuracy in a comprehensive set of scenarios implemented in the DISSECT-CF [1] simulator. With the use of these scenarios, I present a comparative analysis of this model with a similar state-of-the-art simulator. This analysis reveals a significant up to 42.5% improvement in accuracy for modelling data centre energy consumption.

Chapter 1

Introduction

1.1 Motivation

Cloud computing has recently emerged as a paradigm where users rent computational power, hosted by data centres of specialised providers, according on their occasional needs. To host such computational power, Cloud providers are interested in maximizing their profit, either by selling more computational power or by reducing the management and ownership costs. Among these costs, energy consumption is becoming increasingly more important for Cloud providers. For this reason, Cloud providers are currently putting increasing efforts in reducing energy consumption of their data centres.

Energy consumption of data centres can be reduced either by employing energy efficient hardware, or by improving Cloud infrastructure resource management to increase hardware utilisation. Due to the difficulty of upgrading the existing hardware in Cloud infrastructures, Cloud providers are more interested in energy efficient resource management. This encloses many different techniques, such as employing energy-saving features already implemented in the physical machine (PM), increasing hardware utilisation and using virtual machine (VM) consolidation.

Among these techniques, the latter allows mapping of the data centre's VMs on a reduced subset of PMs, in order to allow the data centre manage-

ment to (1) increase utilisation of this subset of PMs and (2) increase energy efficiency of the data centre by shutting down PMs that are not used. Such technique is widely used in modern data centres, as it does not involve any modification on the used hardware (e.g. upgrading the existing hardware, network reconfiguration).

Since this technique is widely used for energy saving in data centres, research in this area may be of interest not only for academia but may have also an impact on industry, due to the interest of Cloud providers in reducing energy consumption of data centres. Research may cover different topics, such as prediction models for energy consumption of VM consolidation and of the benefits that may bring on the long term.

In this work, I focus on providing a simulation framework for the evaluation of VM consolidation algorithms. The main target of this study is the VM migration, which is the main activity involved during the VM consolidation process. Other works have provided a model of energy consumption of VM migration, but at the time of writing, none of them has investigated the impact of VM migration on actors involved in it, such as VMs, PMs and network hardware.

Therefore, in this work I perform an extensive study of VM migration, starting from the energy consumption for transferring VMs over the network, arriving to the study of VM migration energy consumption and its impact on data centre and how to use this model in a Cloud simulation framework.

In the next sections, I describe the main motivations of this work, the problems I found and the way I addressed each one of them, as well as a discussion of the improvements that my work brought to the state-of-the-art.

1.1.1 VM consolidation

According to [2], PMs in data centres are often underutilised. Therefore, improving the utilisation of PMs can improve energy efficiency of data centres. With VM consolidation, it is possible to either (1) increase utilisation of data centre's PM and (2) increase energy efficiency, allowing the data cen-

tre management to shut down the unused PMs. VM consolidation is often described as a vector bin-packing problem [3]. As this problem is known to be NP-complete, its approximated optimal solution is usually obtained through heuristics. There may be different optimization objectives, such as resource utilisation [4, 5, 6, 7, 8], network overhead [9, 10] and energy consumption [11, 12]. Concerning energy consumption, VM consolidation needs especially to (1) understand when is the right time to perform a consolidation (e.g. detecting underutilisation of servers), (2) understanding which remapping improves the current energy efficiency on a long term and (3) determine the energy consumption of VM consolidation. While for the first one software monitoring tools are available, the others are very hard to solve, due to the heterogeneity of architectures inside the data centres and the dynamism of its environment, that makes energy consumption predictions difficult on the long term. In this work, as a step towards this challenging goal, I focus on improving the accuracy of these predictions. To this end, I focus on one of the main activities of VM consolidation: VM migration. As this activity is widely used in VM consolidation, increasing the accuracy of its prediction increases also prediction of energy consumption of VM consolidation, with great advantages for the scientific community working in distributed systems.

1.1.2 VM migration

In order to assess whether a new mapping of VMs is beneficial energy-wise, prediction models are needed for their energy consumption. Such models should take into account all actors (e.g., VMs, PMs, network hardware) and activities (e.g., VM migration, powering off PMs) of VM consolidation. Among all activities, VM migration [13] is one of the most widely used, as it provides the capability of moving the state of running VMs between PMs, thus allowing to dynamically adjust the data centre's workload.

However, VM migration has an energy consumption, as shown by [14, 15]. This consumption must be considered when calculating energy consumption of VM consolidation. Despite having a considerable impact on energy consumption [16], this activity has usually not been taken into account when

performing energy consumption simulations of data centres.

In recent years, several works modelled the energy impacts of VM migration. For example, [17, 18, 19, 16] proposed different models for VM migration energy consumption. Other works tried to consider VM migration when performing dynamic VM consolidation, such as [20, 21, 22]. However, all these works focused only on VMs' workload and have not considered other actors that are relevant for the VM migration process. In fact, VM migration is an activity that involves many different actors, and each one has a different impact on its energy consumption. For this reason, there is significant room for improvement in current VM migration energy consumption models. In fact, different data centre actors are involved in VM migration, like the PMs performing it, the network infrastructure over which the transfer is performed and so on. For this reason, it is reasonable to assume that each time a VM migration is issued, each of these actors is affected energy-wise. Therefore, my goal is to answer the following questions: *How much is the impact of VM migration on energy consumption of data centres? How much is energy consumption of VM migration affected by each one of these actors?*

1.1.3 Network transfer

A large amount of research [23, 24, 25] already focused on reducing energy consumption in data centres and improving their efficiency. Many of these works focus on specific hardware components such as CPUs, storage, memory, but few of them focus on network transfers. In the networking area, existing works investigate energy-saving techniques like sleeping and rate adaptation [26] with focus on routers and switches [27] or on MPI parallel scientific applications [28, 29]. Several works like [30] focused on the energy consumption of network transfers in message passing models, but few investigated it at the *software level*, comprising their complete stacks with different power characteristics and the way they impact the energy consumption of applications. Since data centres often install multiple network interface cards (NICs) on each node, investigating and comparing them at the software level has high potential to enhance the energy efficiency of applications on Cloud

infrastructures.

The network has great importance in this work, for two reasons: first, remapping VMs on different PMs causes also a remapping of the network communications between the VMs, with a remarkable impact on energy consumption of network devices. Moreover, as moving the state of the VMs is done over the network, it is not possible to provide a precise model of VM migration without providing a model of energy consumption of network transfers. Such modelling is made more difficult by the fact that in data centres there are different types of network hardware that are employed, and each one of them may need different modelling for energy consumption. Moreover, measuring energy consumption of the hardware involved in a network transfer (e.g. NICs, routers, switches) is not trivial. For this reason, despite of its impact on data centre energy consumption, very few work tried to model energy consumption of network transfer, or tried to identify the factors that are more impacting for energy consumption. In my research, I attempt to answer the following questions: *How much the network transfers impact on data centre energy consumption? What are the most energy impacting factors for network transfers? Will the use of different network devices have an impact on energy consumption of VM migration?*

1.1.4 Simulation framework

Another important area of interest for this work is the simulation of the Cloud. Due to the difficulty of managing and accessing the internals public Cloud infrastructures, many scientists in the distributed systems community resort on Cloud simulators to validate their results. Since most of the existing research in the energy aware Cloud computing area is related to VM consolidation and VM migration, increasing their modelling accuracy in existing simulators can be of big interest for the research community, allowing researchers to provide more effective energy-aware consolidation algorithms and to test them in a simulation environment that is closer to the real world. Such model should be able to target the complexity of the Cloud and able to consider (1) different architectures, (2) different data centre network/topolo-

gies and (3) different VM migration scenarios. Some efforts in this direction have been already put by the existing Cloud simulators. For example, in recent years, several simulators implemented models for VM migration. For example, the work in [31] added a model to the SimGrid [32] simulator that focuses on the migration's performance overhead, but not the energy consumption stemming from it. Other efforts like [33] provide limited models that, for example, consider only CPU load or network transfers. Moreover, many simulators assume that VM migration does not consume any energy and omit the modelling of its effects on data centre's performance, despite being widely used. For this reason, adding an energy consumption model of VM migration is a significant improvement to the current state-of-the-art in data centre energy consumption simulations. Therefore, because of these problems and the ones that have been highlighted in the previous sections, there is big room for improvement in existing simulators. Therefore, I attempt to answer to the following questions: *How much is it possible to improve the current state-of-the-art in Cloud simulations by considering the parameters identified in this work? What is the benefit of doing this?*

1.2 Objectives

In this section I describe the main objectives of this work.

1.2.1 VM consolidation

The main objective of this work is to provide a theoretical and software framework to develop more accurate energy-aware consolidation algorithms. To this end, I identify which activities are mostly involved in the VM consolidation process, as well as the actors that are impacting its energy consumption. Then, I provide a model for energy consumption of these activities, that will help researchers in predicting energy consumption of a VM consolidation process. Finally, such models will be implemented in a Cloud simulator, to be used for evaluation of energy-aware VM consolidation algorithms. I describe each point in detail in the following sections.

1.2.2 VM migration

I aim at building a energy consumption model for VM migration. Such model should take into account both PM's overcommitment and different types of workloads, running on each actor involved in VM migration. To this end, I perform an investigation on the actors involved in the VM migration and see (1) how much their load impacts energy consumption of VM migration and (2) how much VM migration impacts their energy consumption. From this initial analysis, I build a model for VM migration and find the model coefficients through well-known machine learning techniques. To use machine learning techniques, I need real measurements to build a training and test set for the model. I collect them by designing benchmarks, mimicking real world applications and stressing different actors and hardware components, like CPU and memory. Then, I measure energy consumption of VM migration on different hardware and use these measurements as input for the selected machine learning algorithm. Afterwards, I evaluate the accuracy of my model on different hardware and compare it to the state-of-the-art, showing that my approach increases the accuracy of prediction given by existing models.

1.2.3 Network transfer modelling

I have two objectives in the area of network transfer modelling. First, I want to give an in-depth analysis on the energy impacting factors for data transfers over network. Such investigation will help in understanding which kind of data transfer is more suited to a given interface energy-wise, allowing users to choose the network interface that is best suited for the application he or she plans to run. To this end, I develop a set of benchmarks mimicking different types of network transfers and execute them on different network interfaces. Each benchmark aims at investigating the effects on energy consumption. While executing them, I measure energy consumption of the PMs and see if the identified factor needs to be considered in the modelling, by looking at their impact on energy consumption.

Second, I plan to build a model of energy consumption of network transfer. Such model should be able to predict the energy consumption on both

network interfaces I targeted in my study and work with different types of network transfers. Since in data centres an increasingly bigger amount of data is transferred over the network, having such a model for energy consumption of network transfer can help to increase accuracy of energy consumption prediction in data centres. I develop this model starting from measurements collected during the benchmark execution in the DPS Cloud and using supervised learning techniques on the measured data.

1.2.4 Simulation

Once I build my network transfer model, I want to use it to improve accuracy of existing Cloud simulators. Because of the difficulty in managing and accessing data centre infrastructure, many researchers in the distributed computing community resort to use Cloud simulators. However, such simulators either do not provide adequate support for prediction of energy consumption or there is big room for improvement in the accuracy of their predictions. I plan to extend an existing simulator with my model for network transfer and my model for VM migration. Afterwards, I plan to evaluate the accuracy of the modified simulator with the state-of-the-art, showing that by using my models for network transfer and VM migration I am able to improve accuracy of existing simulators, bringing an important advantage to scientific community.

1.2.5 Summary

These objectives aim at improving accuracy of data centre simulations and to provide tools to support energy saving decisions in data centres. Since the target scenario of my work is VM consolidation, I focus on its main activity, that is VM migration. To build a model for VM migration, I first model it as a network transfer, therefore building and evaluating a model for energy consumption of network transfers. Afterwards, I investigate the parameters that are impacting the most the energy consumption of VM migration and build on the top of my network transfer model an energy consumption model for VM migration. This model takes into account not only the network

transfer part, but also different other parameters that are of interest for its modelling. Afterwards, I implement my models into an existing Cloud simulator and show that using them the accuracy of energy consumption prediction increases. At each step, I perform a detailed analysis on how the proposed model is able to provide more accurate estimations compared to the state-of-the-art. Therefore, I carefully evaluate the accuracy of my model by comparing it with real world measurements, and, wherever possible, with existing energy models, to show an improved accuracy.

1.3 Outline

In this section I describe the organisation of my work.

In Chapter 2 I introduce the theoretical foundations of this work and describe all the important terms and technologies used in the thesis. I define all the actors involved in the VM migration and describe the different VM migration approaches considered in this work and the way I model it. I also describe the network transfer model and the parameters that are more energy-impacting for its consumption.

Afterwards, I describe in Chapter 3 the benchmarking methodology used in this work. First, I describe the framework I employed to collect the energy measurements that are necessary for my work. Then, I describe the applications used on the PMs to simulate different types of load, both for VM migration and network transfers.

Then, I perform in Chapter 4 the validation of my network transfer model by comparing its predictions to the real measurements and showing its accuracy. Then, I use the same model on a small subset of measurements for VM migration and discuss the accuracy of this model also in this scenario.

Afterwards, I validate in Chapter 5 my VM migration model using the measurements collected with the benchmarking methodology described in Chapter 3. I describe how I use these measurements as training and testing set for linear regression to build a VM energy consumption model. I compare the accuracy of this model with other existing models, showing that my model is able to increase the accuracy of prediction by 24%.

Chapter 6 describes how my findings are integrated in a simulation framework. First, I show that the GroudSim/DISSECT-CF simulation framework is the most suited to perform my extensions. Therefore, I describe the simulation framework in detail, explaining how I extend it to include my work. Then, I show the correctness of this implementation by comparing results obtained with my simulation with the real world measurements. Afterwards, I perform a comparison with CloudSim state-of-the-art simulator, showing that my model can provide an improvement of up to 45.6% in accuracy of energy consumption prediction.

Finally, I conclude my thesis in Chapter 7, describing its contributions and outlining the future work.

1.4 Related Work

In this section I present the related work, divided according to the research areas that are covered in this work.

1.4.1 Data centre modelling

In this section I describe the related work about implementing an accurate Cloud simulator. A Cloud simulator should provide simulations at three level: the physical level, concerning the PMs and the network infrastructure. Then, the virtualization layer above them that provides additional functionalities (VM migration, provisioning and placement). I analyse all of them in details next.

PM modelling

In this section I outline how existing simulators handle the simulation of the physical infrastructure. One of the first works trying to understand energy consumption of data centre is [34] but do not try to model it. Most of the existing work simulate the behavior of physical infrastructure according to CPU, memory, storage and network. I analyse each one of them in detail.

Concerning CPU modelling, existing papers either focus on a specific CPU architecture [35, 36, 37] or assume a linear relationship between CPU usage and energy consumption [38, 39], which may lead to inaccurate results [40]. Moreover, most of this models do not consider the virtualization overhead, making it not suitable for virtualized data centre.

Concerning memory modelling, works like [41, 42] provide simulations of DRAM behaviour, but neither the virtualization overhead nor the energy consumption is considered. Works like [43] provides modelling of memory resource management in VMWare ESX Server, but no energy modelling is provided. Similar works like [44, 45] provide insights about memory management, respectively for Xen and KVM hypervisor.

Storage energy modelling has been provided by [46] and other similar works like [47, 48, 49]. These works, however, do not consider the virtualization overhead, neither distributed storage systems.

Data centres network performances has been instead modelled in works like [50]. Other works like [51] propose data centre network simulators, without considering energy consumption. Works like [52, 53] target how to improve energy efficiency of networking in data centres, while works like [54, 55] try to model energy consumption of network transfers, but do not use this model in simulations.

Works like [56, 57, 58] provide a joint approach to data centre modelling. However, all these works have the problems previously outlined. Moreover several features provided by the virtualization layer available in data centres are not modelled. I analyse them in details in the following sections.

1.4.2 Network transfer modelling

Energy aware networking

Many works exploit network awareness to save energy, with focus on routing equipment and algorithms: In [59] energy-aware allocation of resources in Clouds considering network topology is investigated, while [60] proposes a network power manager which dynamically manages routers to reduce energy consumption and [61] proposes a model for energy-aware routing. The

paper [62] investigates client-centered techniques for energy efficient communication on IEEE 802.11b networks, which interest more wireless communication than data centers. Complementary to these works, I focus on the energy consumption from the perspective of software application, including not only the NICs, but also the other components involved in network transfers.

Network energy modelling

One of the first studies on network energy consumption focuses on energy consumption of routers, switches and hubs [63] but does not take into account energy consumed by the NICs. Many works like [27, 64] provide models for router power consumption, but do not consider the power consumed by NICs for network transfers.

Other works like [65], [66] provide models for the energy consumption of wireless network interfaces, which are of interest to mobile devices rather than data centres. [67] and [68] propose other energy consumption models, the former for optical IP networks and the latter for network-on-chip routers. In [69] a energy consumption model for network equipment and transfers for large-scale networks, based on transfer time and bandwidth, is introduced. In this thesis I propose a complementary model for network transfers considering different NICs and more parameters. Works like [26] consider only transfer time when building a model for network transfers. In this work, other additional factors are considered.

1.4.3 VM migration modelling

Live VM migration has been proposed by [13] for the Xen hypervisor. Since then, it has been implemented in many popular hypervisors, such as Xen, KVM and VMWare. Many works like [70, 71, 72, 73] exploit live VM migration to perform energy-aware VM consolidation. However, energy consumption of VM migration is not taken into account in these works. Other works like [74, 75, 76] focused on the cost of live migration for Cloud data centres, but considered only performance and did not take energy consumption into account. Further works like [31] implemented a model for VM migration in

a Cloud simulator, but do not provide models for its energy consumption. Recent works like [77] consider the time of live migration, but this study consider only CPU-intensive workloads and does not take energy into account. Other works like [78] propose a probabilistic approach to quantify the cost of VM live migration, but this cost does not take energy into account. First investigations about energy consumption of VM migration have been done by [79]. One of the first works that modelled time, energy and performance of live migration at the same time is [17], which identified a relationship between network bandwidth and energy consumption of Xen live migration. This work, however, considers only the load running on the migrating VM and makes the simplistic assumption that source and target host have the same energy consumption for VM migration. A similar work has been done for KVM live migration by [18]. Another model has been proposed by [19], but this model considers only CPU load.

In this work, I consider the workload of each actor involved in the migration process and extract a more accurate model for both live and non-live VM migration.

1.4.4 Cloud simulators

In this section I describe the state-of-the-art simulators and try to outline their pros and cons.

A Cloud simulator should be able to simulate different levels of a Cloud: the *infrastructure* layer and the *data centre management* layer.

At the infrastructure level, a simulator should be able to simulate both the (1) PM behaviour, with all its subsystems and (2) the behaviour of the virtualization layer, added to support VMs or containers. At this layer also energy consumption simulations should be added, to address the growing interest of the distributed systems research community in energy consumption. At the Cloud management layer, further activities, like VM placement, VM migration and VM provisioning, as well as physical infrastructure management operations like shutting down/turning on the PM. CloudSim [33] is the most used and cited simulator of different components of the data cen-

tre infrastructure, including internal networking and energy consumption. Moreover, it does not take into account several important parameters in its VM migration model such as overcommitment and memory dirtying rate.

SimGrid [32] provides a scalable and fast simulation framework of Cloud data centres, Grid and peer-to-peer systems, including a model for simulating VM migration [31]. However, it provides no energy consumption model for VM migration (at the time of writing).

GreenCloud [80] offers packet-level simulations for energy-aware Cloud computing data centres. It provides the capability of separately modelling the energy consumption of all data centre components, including CPU, network, and storage. However, its CPU model is based on Xeon processors only and no energy consumption model for live migration is provided.

DCSim [81] provides fine-grained data centre simulation at different levels of abstraction considering networking and energy consumption models, with no support for VM migration.

GroudSim [82] is the simulation backend of the ASKALON system [83] that, due to its integration with the DISSECT-CF [84] Cloud infrastructure simulator, provides models for energy consumption of data centre components, as well as for VM migration and networking.

iCanCloud [85] provides an easily extensible simulation framework for Infrastructure as a Service (IaaS) Clouds. It provides the possibility to simulate the behaviour of CPU, memory, storage and network subsystems. However, it does not offer the possibility of simulating energy consumption of a data centre at the time of writing.

Chapter 2

Model

2.1 Introduction

In this section I introduce all the terms that are necessary to understand the presented topics. First I give definitions of the actors that are important for this work. Then, I focus on VM migration process, describing the actors involved in this activity and its power characteristics. After this analysis, I design a model of data centres energy consumption that takes into account all the hardware components affecting it. Then, I design a model for network transfers, from which I derive a model for VM migration. The model validation is described in Chapters 4 and 5.

2.2 Definitions

In this section I describe the models of the data centre components that are related to this work.

2.2.1 Data centres

Data centres are the cornerstone of Cloud computing. A data centre is a facility hosting the computational power that is rented to the users of the Cloud. Data centres are highly dynamic environments, therefore their configuration may vary over time, according to the amount of load and the

available resources. Therefore, a data centre can be seen as a set $\mathcal{D}$ defined as follows:

Definition 1. *A **data centre**, is a vector $\mathcal{D}$,*

$$\mathcal{D} = [\mathcal{H}, \mathcal{V}, \mathcal{V}^*(\mathcal{H}), \mathcal{N}], \tag{2.1}$$

where:

- *$\mathcal{H}$ is the set of PMs inside the data centre $\mathcal{D}$ during its whole lifetime. PMs are defined in Definition 3;*
- *$\mathcal{V}$ is the set of VMs inside the data centre $\mathcal{D}$ during its whole lifetime. VMs are defined in Definition 5);*
- *$\mathcal{V}^*(\mathcal{H})$ is a set of pairs (v, h) defining the initial allocation of the VMs on PMs, where $v \in \mathcal{V}$ and $h \in \mathcal{H}$. I can define it as $\bigcup_{h \in \mathcal{H}} \mathcal{V}^*(h)$, defined in Definition 3;*
- *$\mathcal{N}$ is the set of all the network switches inside the data centre. Switches are defined in Section 4).*

During the data centre's lifetime, modifications on the data centre state may occur (e.g. startup/shutdown of VMs and PMs, VMs' migrations). Because of many factors, such as data centre energy saving policies, hardware failures, and users' demand, the amount of available VMs at a given instant of time may vary. Finally, each time a VM consolidation is performed, the mapping of VMs to PMs changes. Therefore, I define the state of the data centre $\mathcal{D}$ at the instant t, namely $\mathcal{D}(t)$, as follows:

Definition 2. *The **state of a data centre** $\mathcal{D}$ at the instant t is defined as a vector $\mathcal{D}(t)$,*

$$\mathcal{D}(t) = [\mathcal{H}(t), \mathcal{V}(t), \mathcal{V}^*(\mathcal{H}, t)], \tag{2.2}$$

where:

- *$\mathcal{H}(t)$ is the set of PMs that are available at instant t. Clearly, $\mathcal{H}(t) \subseteq \mathcal{H}$;*

- $\mathcal{V}(t)$ *is the set of VMs that are running at instant* t*. Clearly,* $\mathcal{V}(t) \subseteq \mathcal{V}$*;*
- $\mathcal{V}^*(\mathcal{H}, t)$ *is a set of pairs* (v, h) *defining the allocation of the VMs on PMs at time* t*. Clearly,* $h \in \mathcal{H}(t)$ *and* $v \in \mathcal{V}(t)$*.*

In the next section, I define the PMs characteristics.

2.2.2 Physical machine

A PM is an hardware device, such as a personal computer or any other computing device. PMs are used to host the VMs that execute the computation. Resources offered by the PMs are shared between all the VMs that are hosted by the PM. The sharing of the resources is managed by the hypervisor, a program running on the PM operating system (OS), or even a modified version of a PM's OS (e.g. Xen), that is responsible for allowing the different OSs running on the VMs to use the PM's resources. A PM can have a diverse amount of hardware resources. In this work I mostly focus on CPU, RAM, storage and network. For this reason, a PM $h \in \mathcal{H}$ can be seen as a vector of its amount of resources and the set of VMs that are allocated on it.

Definition 3. *I define a* ***physical machine*** h *as follows:*

$$h = [\texttt{CPU}_{max}(h), \texttt{RAM}_{max}(h), \texttt{BW}_{io}^{max}(h), \texttt{BW}_{net}^{max}(h), \mathcal{V}^*(h), s], \tag{2.3}$$

where:

- $\texttt{CPU}_{max}(h)$ *is the maximum CPU load that is possible to allocate to host* h*;*
- $\texttt{RAM}_{max}(h)$ *is the maximum amount of RAM for host* h*;*
- $\texttt{BW}_{io}^{max}(h)$ *is the maximum I/O bandwidth;*
- $\mathcal{V}^*(h)$ *is the set of VMs that are allocated to host* h*.*
- s *is the network switch to which* h *is connected (see Definition 4).*

From now on, $\texttt{CPU}_{max}(h)$ is defined in MIPS (Millions of Instructions Per Second), $\texttt{RAM}_{max}(h)$ in bytes and the bandwidth in bytes per second. It must be pointed out also that $\texttt{CPU}_{max}(h), \texttt{RAM}_{max}(h), \texttt{BW}_{io}^{max}(h), \texttt{BW}_{net}^{max}(h)$ do not necessarily reflect the amount of CPU, RAM, and bandwidth available on PM h, because in some cases the data centre may allow overcommitment.

Definition 4. *A **switch** s is defined as:*

$$s = [\texttt{BW}_{net}^{max}(s), \mathcal{H}_{(s)}, \mathcal{N}_{(s)}], \tag{2.4}$$

where:

- $\texttt{BW}_{net}^{max}(s)$ *is the bandwidth of the switch s;*
- $\mathcal{H}_{(s)}$ *is the set of PMs, defining the hosts that are connected to s;*
- $\mathcal{N}_{(s)}$ *is the set of switches, defining the switches that are connected to s.*

Formally, $\exists x \in \mathcal{H}_{(s)} \iff$ switch s connects x and $\exists y \in \mathcal{N}_{(s)} \iff$ switch s connects y. Clearly, $\mathcal{H}_{(s)} \subset \mathcal{H}$ and $\mathcal{N}(s) \subset \mathcal{N}$.

Concerning the networking, I consider that in modern data centres there are many types of LAN technologies available. Such new technologies are different from the typical LAN technologies and offer an higher latency and throughput, to meet the growing needs of modern Cloud applications. Such technologies may need to rely on a different physical layer technology, different from the typical CSMA/CD used by Ethernet. Among them, the most successful one is Infiniband, that is based on the RDMA technology. I will later on describe both technologies in detail.

2.2.3 System load model

I assume that power consumption of each component is proportional to its load. Therefore, before defining a model for power consumption of each

component, I need to define a model for the load of each component. I calculate the load of CPU resources as follows:

$$load_{cpu}(h,t) = \frac{\texttt{CPU}(h,t)}{\texttt{CPU}_{max}(h)}, \tag{2.5}$$

where $\texttt{CPU}(h,t)$ is the amount of MIPS used by the PM h at time instance t. I also define the network load as:

$$load_{net}(h,t) = \frac{\texttt{BW}_{net}(h,t)}{\texttt{BW}_{net}^{\max}(h)}, \tag{2.6}$$

where $\texttt{BW}_{net}(h,t)$ is the bandwidth available on host h at the time instance t. $load_{net}(h,t)$ can be seen as a measure of how many bytes are sent/received over the network at the time t on PM h. Finally, I define the disk load as:

$$load_{io}(h,t) = \frac{\texttt{BW}_{io}(h,t)}{\texttt{BW}_{io}^{\max}(h)}, \tag{2.7}$$

where $\texttt{BW}_{io}(h,t)$ is the bandwidth available at the time instance t. $load_{io}$ can be seen as a measure of the amount of bytes that are read/written at time t.

2.2.4 Virtualization

Modern Cloud data centres rely on a technology called virtualization. By virtualization I refer to the creation of a virtual instance of a resource such as a PM, storage or network. In the Cloud scenario, the resource that is virtualized is the OS that is running the computation, requested by the users of the infrastructure.

Virtualization is offered by different softwares, called hypervisors. Currently, the most used one are Xen [86], KVM [45] eSXi and VMWare server, used by VMWare [87] virtualization platform. There are different types of OS-based virtualization services, like:

- *Kernel-based virtualization*, in which virtualization is provided by exploiting kernel extensions to the PM's OS. The VM OS (guest OS) communicates with the kernel of the PM's OS, that provides the access

Hypervisor	Kernel-based	Hardware-assisted	Paravirtualization
KVM	✓	✗	✗
Xen	✗	✓	✓
eSXi	✗	✓	✗
VMWare server	✗	✓	✗

Table 2.1: Hypervisors summary.

to the underlying hardware.

- *Hardware-assisted virtualization*, in which the virtualization features required by the guest OS are provided by hardware extensions. The guest OS communicates directly with the hardware to obtain what is needed by the VM. This type of virtualization must be explicitly supported by the CPU.

- *Paravirtualization*, in which the hypervisor provides to the guest OS an API to communicate with the PM's hardware. The guest OS must be accordingly modified to exploit the API provided by the hypervisor.

In Table 2.1 I summarize the different type of virtualization according to the different hypervisors. According to the hypervisor and the type of virtualization, there is a different type of virtualization overhead, as well as a different use of physical resources. This results in a different PM's energy consumption. For this reason, In my model I also include the power consumption for managing the virtualization. In this work, I focus on paravirtualization, as it ensures the less virtualization overhead at the time of writing.

There are also two other things to be considered, due to their impact on both data centre energy consumption and performance: resource overcommitment and VM migration. In this work, I focus mostly on VM migration, as it has the highest impact in energy consumption of VM consolidation. However, for defining a VM migration, it is necessary to first define a VM. I do this in the next section.

2.2.5 Virtual machine

Virtual machines (VMs) are the basic computing entities of Cloud computing, as they are rented to Cloud users to perform their computation. Each VM can be either specifically tailored for the user who rents it from the Cloud provider or an instance of a pre-defined image offered by the provider. A VM can be customized in terms of different parameters, such as CPU power, RAM and storage amount as well as the OS that can be executed on it. The parameters that are relevant for this work are CPU, RAM, network and I/O bandwidth. Therefore, I define a VM v as follows:

Definition 5. *A **VM** v is a vector*

$$v = [\texttt{CPU}_{max}(v), \texttt{RAM}_{max}(v), \texttt{BW}_{io}^{max}(v), \texttt{BW}_{net}^{max}(v)], \tag{2.8}$$

where:

- $\texttt{CPU}_{max}(v)$ *is the maximum CPU available on VM* v*;*
- $\texttt{RAM}_{max}(v)$ *the maximum amount of RAM available on VM* v*;*
- $\texttt{BW}_{io}^{max}(v)$ *is the maximum amount of storage bandwidth available on VM* v*;*
- $\texttt{BW}_{net}^{max}(v)$ *is the maximum amount of network bandwidth available on VM* v*.*

The VMs use the resources of the PMs on which they are allocated. In order to provide the computational resources that are used by the data centre users, each VM needs to be allocated to a host, namely, being added to $\mathcal{V}^*(h,t)$ (see Definition 3). To be allocated to host h, $\mathcal{V}^*(h,t)$ must respect the following property:

$$\sum_{v \in V^*(h,t)} [\texttt{CPU}_{max}(v), \texttt{RAM}_{max}(v), \texttt{BW}_{io}^{max}(v), \texttt{BW}_{net}^{max}(v)] \leq$$
$$[\texttt{CPU}_{max}(h), \texttt{RAM}_{max}(h), \texttt{BW}_{io}^{max}(h), \texttt{BW}_{net}^{max}(h)] \tag{2.9}$$

which means that the sum of all the CPU and RAM resources of all the VM allocated to host h must not exceed the CPU and RAM available on the host. For what concerns networking, I assume that the VMs uses a portion of the PM's bandwidth, namely $\texttt{BW}_{net}^{max}(v)$, and that the networking and connectivity are provided by the PM that hosts v. In modern data centres is possible also to set virtual networks between VMs, adding another level of network virtualization, but I do not consider it in this work, as it has no impact on energy consumption of VM migration.

2.3 VM migration

In this section I provide an overview of the power characteristics of VM migration. First, I describe the VM migration process and then the actors involved in this process. Afterwards, I investigate the workloads impacting the energy consumption of VM migration and finally, I identify the phases that occur during a migration.

2.3.1 Preliminaries

I provide here a complete discussion of migration process, in order to better understand its power characteristics. VM migration is an activity that changes the state of a data centre $\mathcal{D}(t)$, defined in Definition 2. The change in $\mathcal{D}(t)$ is reflected by the modification of the content of the set $\mathcal{V}^*(\mathcal{H})$, since VM migration moves the VM from one PM to another. A VM migration happens between two PMs, namely the source and the target host (respectively, the one from which the VM migration is issued and the one that runs the VM after the completion of the VM migration, denoted respectively as $\mathcal{S}$ and $\mathcal{T}$). I define the source host as $\mathcal{S}$ and the target host as $\mathcal{T}$. At the time instant t, $\mathcal{S}, \mathcal{T} \in \mathcal{H}(t)$. When migrating a VM v from $\mathcal{S}$ to $\mathcal{T}$, it happens that at an instant t the mapping of the VM v changes from PM $\mathcal{S}$ to PM $\mathcal{T}$, formally $v \in \mathcal{V}^*(\mathcal{S}, t-1), v \notin \mathcal{V}^*(\mathcal{T}, t-1)$ and $v \notin \mathcal{V}^*(\mathcal{S}, t), v \in \mathcal{V}^*(\mathcal{T}, t)$. In the next section, I describe how the VM migration can be performed.

VM migration approaches

Although VM migration can be realised in different ways, I focus here on the most used approaches: non-live migration and live migration. I analyse them in detail now:

Non-live migration (sometimes referred as *suspend-resume migration*) approach consists of: (1) suspending the VM to be migrated, (2) transferring its state to the target host, and (3) resuming the VM on the target host. This approach is the least energy impacting, because of the fact that the migrating VM is suspended during all the time of the migration. For this reason, energy is consumed just to save the state of the VM on the source host, transfer it over the network and then resume it on the target host.

Live migration has been proposed to reduce the downtime of the VM during migration. The idea is to move the state of the VM while the VM is still running, keeping the state of the VM consistent on both $\mathcal{S}$ and $\mathcal{T}$ while performing the migration. There are two types of approaches to perform live migration: pre-copy live migration and post-copy live migration. I describe both approaches in the following:

Pre-copy migration proposed by [13] consists of six steps:

1. moving the VM state from source to target host while the VM operates normally;
2. updating the state of target host with the modifications occurred on the source during state transfer;
3. repeating step (2) until a predefined termination criteria is reached (e.g., the size of the VM state difference reaches under a given threshold or maximum number of updates reached);
4. suspending the VM and transferring its last state changes to the target; (5) resuming the VM on the target when its state is consistent with the source;

5. destroying the suspended VM on source.

This approach reduces the gap in VM lifetimes, ensuring an higher availability of the VM, but consumes more energy for two reasons. First, the migrating the VM is still running, thus the migration energy consumption involves not only the cost of transferring VM state but also the energy consumption of the VM while running. Second, if the state on the source changes faster than it can be transferred over network to the target, the updates to the VM's state has to be continuously transferred, considerably increasing the VM migration time and, consequently, its energy consumption.

Post-copy migration proposed by [88] consists of five steps:

1. moving the VM CPU state from source to target host. During this time, VM is suspended;

2. resuming execution of the VM on the target host;

3. sending the VM memory pages to the target host from source host while the VM is running on the target host;

4. for each page fault experienced on the VM in the target host, actively pushing the faulty pages from source host to target, concurrently with step (3);

5. destroying the VM on the source host when all the memory pages have been transferred.

This approach reduces the VM migration time, it makes possible to run the VM on the target host without copying all the VM memory pages, like in the pre-copy migration. However, it may have detrimental effects on the performance of the applications running on the VM, as each page fault may result in a network transfer from source to target host, especially for applications that continuously update a big amount of memory pages inside the VM. This may also result in a higher energy consumption, due to the continuous sending of memory pages over the network.

In this work, I consider both non-live and live migration. For live migration, I only consider pre-copy migration, as at the time of writing no commercial hypervisor uses post-copy migration. For this reason, I choose to focus only on pre-copy migration, as it allows me to validate my results on real-world implementations offered by commercial products, like Xen. Therefore, from now on, whenever I refer to live migration, I imply a pre-copy VM migration.

Actors

After describing the VM migration process and its phases, I identify in this phase the actors involved in the VM migration process, as detailed in Figure 2.1. I assume that in each data centre there is a software entity called consolidation manager that constantly monitors the load of the data centre, assumption that is common to many works like [89, 90, 91]. This entity may decide at a certain point to initiate a VM migration. When the migration is issued, the VM v is migrated over the network from $\mathcal{S}$ to $\mathcal{T}$.

At the end of this process, VM is running on the target host. This process is summarised in Figure 2.1, in which the involved actors are highlighted. Therefore, I conclude that these are the actors involved in a VM migration:

Consolidation Manager constantly monitors the load of the data centre, selects the VM to be migrated and the target host, and finally initiates the migration. Afterwards, it returns to its previous operation.

Migrating VM is a VM $v \in \mathcal{V}$, that has to be transferred from $\mathcal{S}$ to $\mathcal{T}$, while the VM is also expected to be running services used by the customers of the data centre or the service provider utilising the VM.

Source host is the PM $\mathcal{S} \in \mathcal{H}$. At this time t, $v \in \mathcal{V}^*(\mathcal{S}, t)$. The source establishes the initial connection with $\mathcal{T}$ through its switch s.

Target host is the PM $\mathcal{T} \in \mathcal{H}$ designated by the consolidation manager as the destination for the migrating VM (i.e., the host that executes the VM

after the migration process completes).

Network refers to the underlying communication infrastructure responsible for connecting the other actors and for supporting the VM state transfer, namely the set $n \subset \mathcal{N}$ of data centre switches that are involved in the VM migration.

In the rest of the thesis I focus only on three of these actors: v, $\mathcal{S}$, and $\mathcal{T}$. I do not consider the consolidation manager because it does not further interact with the migration after initiating it. I also ignore the network infrastructure because it affects the VM migration only at its maximum utilisation and it can be safely assumed that a VM migration is never issued when the bandwidth between two hosts is fully utilised, meaning that, if t is the instant at which VM migration is issued, $load_{net}(\mathcal{S}, t) = BW_{net}^{max}(\mathcal{S})$, $load_{net}(\mathcal{T}, t) = BW_{net}^{max}(\mathcal{T})$ and $load_{net}(s, t) = BW_{net}^{max}(s) \forall s \in n$. Therefore, for the network transfer part I use the model of Section 2.4.2, assuming that the maximum bandwidth is available when the consolidation manager issues a VM migration.

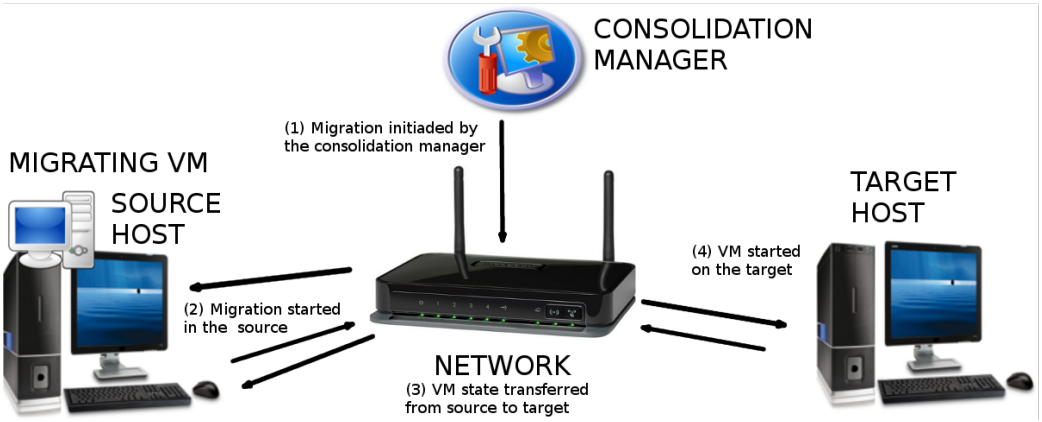
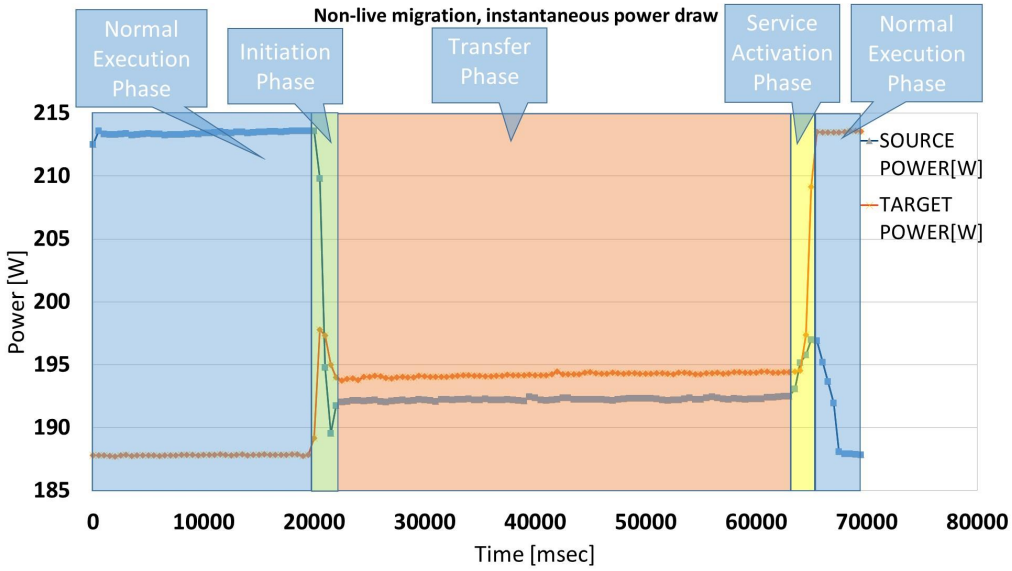


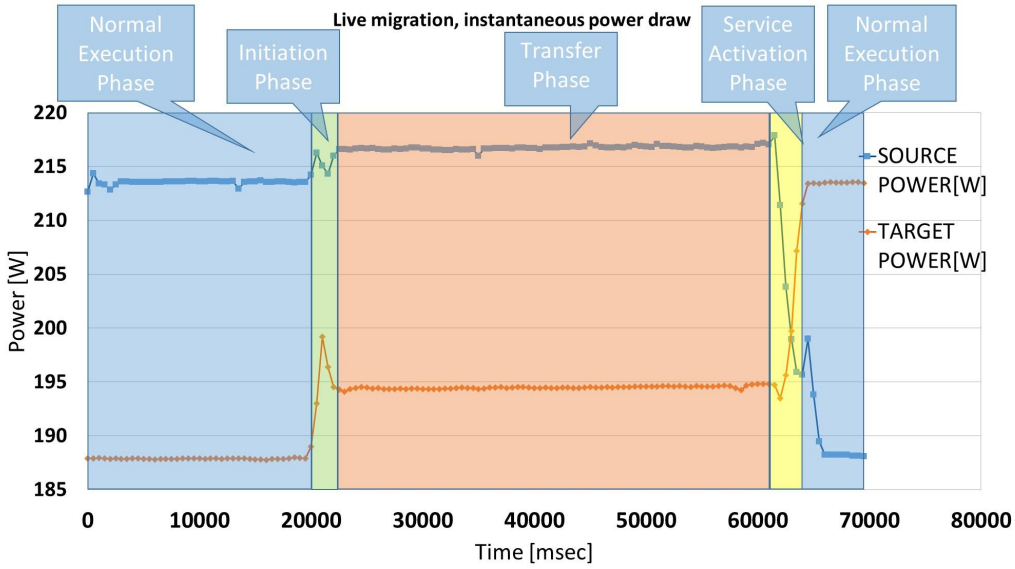
Figure 2.1: Summary of the migration process.

Migration energy phases

As I discussed in the previous sections, both live and non-live migration go through different phases that could lead to different energy-wise behaviour for



(a) Non-live migration



(b) Live migration

Figure 2.2: Energy consumption phases of non-live and live migration.

each actor. In this section, I identify those phases of VM migration that differ from an energy point of view by collecting and analysing instantaneous power draw traces of a VM migration (see the traces and phases in Figure 2.2).

Normal execution During this phase, each actor performs its normal operation as there was no migration decision taken so far. For the sake of simplicity, I ensure a constant energy consumption over this phase so the acquired power traces cannot be accounted for anything else but the migration process discussed in the later phases.

Initiation This phase starts when the migration is requested by the consolidation manager and ends when the target host is prepared to receive the VM state. In case of non-live migration, the source host experiences a strong decrease in power consumption because the migrating VM is suspended in the beginning of this phase. In contrast during live migration, the source host reaches a new peak for energy consumption because of the preparation tasks necessary for sending the migrating VM to the target. The target host shows independent behaviour from the VM migration approach applied. It experiences peaks in its power draw due to checking of resource availability and acknowledging to the source that the migration can start.

Transfer During this phase, all the state information of the VM is transferred over the network from the source to the target host. Compared to the initiation phase, there is an increase of power drawn introduced by the exchanged VM state data in both the live and non-live migration approaches. For live migration, an additional consumption is recorded for the source because it needs to keep track of the modifications to the VM state. During this phase, I can notice a sub-phase that I call downtime phase: during this phase the VM is totally unavailable in both source and target host. It is important to keep track of the energy consumed in this phase, because during the downtime no computation is performed by the services residing on the migrating VM. In the meanwhile, the source is freeing the resource previously owned by the VM and the target is preparing to run the VM. I expect in this phase that the most impacting factors is the CPU load, because it has an impact on the time it takes to run the VM on the target.

Service activation This phase starts after the VM state is transferred and ends when the VM is running on the target host. In this phase, the source host frees the resources that previously belonged to the migrating VM (please note, before freeing up the resources the source host must shutdown the migrating VM in case of live migration). The target host instead runs the VM machine. Finally, each actor returns to the normal execution phase.

2.4 Energy model

In this section, I describe a generic computing oriented (e.g., it does not include air conditioning) data centre energy consumption model underneath my approach, derived from my previous study in [92]. I design this model by first making it able to predict the instantaneous power consumption of a server in a data centre, and then extending it with a model for live VM migration (starting from Section 2.5). This model considers the CPU utilisation as the main parameter of PMs and extracts a regression model for it. Although CPU is the most impacting factor according to [93], I also consider network transfers and storage to improve accuracy over existing models that usually do not consider them, regardless of their impact on the energy consumption.

The computing oriented energy consumption in a data centre E_d is given by the integral of its instantaneous power draw $P_d(t)$:

$$E_d = \int_{t_{start}}^{t_{end}} P_d(t)\ dt, \tag{2.10}$$

where $[t_{start}, t_{end}]$ is the interval for which I calculate energy consumption, and $P_d(t)$ is the sum of the instantaneous power draw of the network infrastructure $P_{inf}(\mathcal{N}, t)$ and the instantaneous power $P(h, t)$ of each PM h at time instant t:

$$P_d(t) = P_{inf}(\mathcal{N}, t) + \sum_{h \in \mathcal{H}(t)} P(h, t). \tag{2.11}$$

In the rest of this work, I consider $P_{inf}(\mathcal{N}, t)$ as a constant, as I expect this power draw to not be of impact for VM migration. Concerning instantaneous PM's power consumption, it is the sum of the consumption of each of its components. While the consumption of some components (e.g. memory, PCI slots, motherboard) is constant over time, some other components consumption (CPU, I/O operations, network) varies depending on the load. Therefore, I distinguish two parts, $P_{idle}(h, t)$ and $P_{active}(h, t)$:

$$P(h, t) = P_{idle}(h) + P_{active}(h, t). \tag{2.12}$$

The idle power $P_{idle}(h)$ is a constant representing the consumption required by the machine just to be on. It includes also the power draw of RAM, that I assumed to be constant. The active power $P_{active}(h, t)$ is instead dependent on the PM subsystems' load. Its value is comprised between 0 and $P_r(h) = P_{max}(h) - P_{idle}(h)$, where $P_r(h)$ is the size of the interval of values in which $P_{active}(h, t)$ is defined. $P_{max}(h)$ is the maximum power consumption on the host h. For simplicity, I assume that P_{active} is influenced mostly by CPU, network and I/O operations, as stated by [94], therefore I define it as follows:

$$P_{active}(h, t) = P_{cpu}(h, t) + P_{net}(h, t) + P_{io}(h, t). \tag{2.13}$$

Where $P_{cpu}(h, t)$ is the power draw of CPU, P_{net} is the power consumption of network and P_{io} is the power consumption of disk operations, that are the components identified in the definition 3. As energy consumption is the integral of power draw, I obtain the active energy consumption of host h, E_h in the following way:

$$E_{active}(h) = \int_{t_{start}}^{t_{end}} P_{active}(h, t)\ dt. \tag{2.14}$$

Then, by applying the linearity of the integral, I obtain the equation of the energy consumption of each component:

$$E_{cpu}(h) = \int_{t_{start}}^{t_{end}} P_{cpu}(h, t)\ dt, \tag{2.15}$$

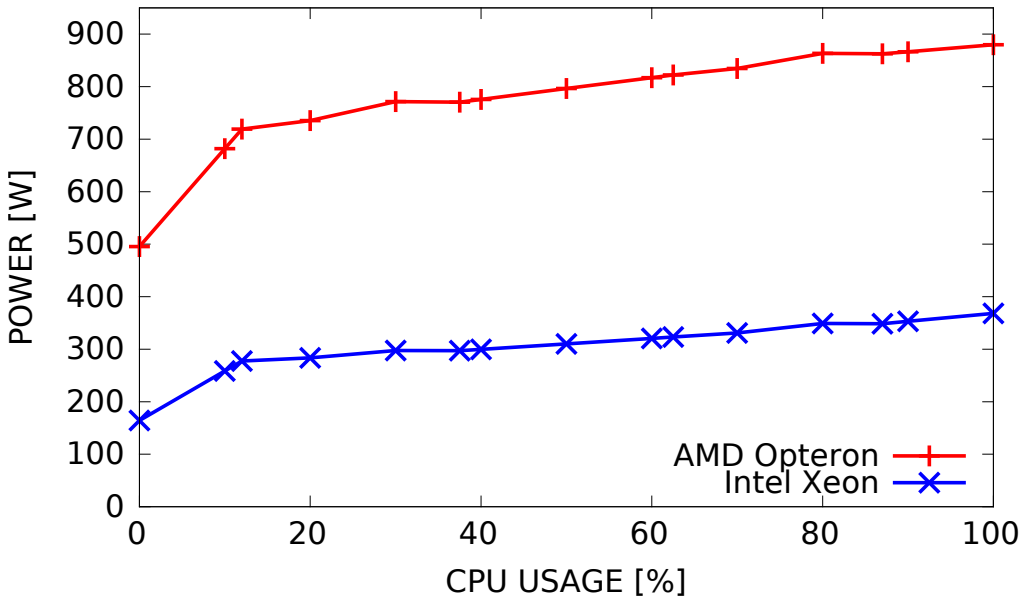


Figure 2.3: Instantaneous power consumption of a host in relation to CPU utilization.

$$E_{net}(h) = \int_{t_{start}}^{t_{end}} P_{net}(h,t)\ dt, \tag{2.16}$$

$$E_{io}(h) = \int_{t_{start}}^{t_{end}} P_{io}(h,t)\ dt. \tag{2.17}$$

I analyse each one of them in the following sections.

2.4.1 CPU power model

As CPU is the most impacting component on the energy consumption of a PM [93], I focus mostly on this model. Most works on energy modelling [95, 96] assume a linear relationship between the CPU use and its power consumption. However, power consumption is more aligned with a piecewise linear trend according to the observations in Figure 2.3 using the two sets of machines presented in Table 3.5c. The initial transition from idle to non-idle state, when several hardware components are simultaneously starting to consume power (e.g. caches), produce in a higher growth in power

consumption at low load levels. Once all components are powered up, the power grows at a different trend:

$$P_{\text{low-cpu}}(h,t) = \alpha(h) \cdot P_r(h) \cdot load_{cpu}(h,t), \tag{2.18}$$

$$P_{\text{high-cpu}}(h,t) = \beta(h) \cdot P_r(h) + (1-\beta(h)) \cdot P_r(h) \cdot load_{cpu}(h,t); \tag{2.19}$$

$$P_{\text{cpu}}(h,t) = \begin{cases} P_{\text{low-cpu}}(h,t), & load_{cpu}(h,t) \leq \mathcal{L}(h); \\ P_{\text{high-cpu}}(h,t), & \text{otherwise}, \end{cases} \tag{2.20}$$

where $\mathcal{L}(h)$ is the load at which the trend changes on host h, $\text{load}_{cpu}(h,t)$ is the CPU load on host h at time instance t, and $P_r(h) = P_{\max}(h) - P_{\text{idle}}(h)$, where $P_{\max}(h)$ and $P_{\text{idle}}(h)$ are the maximum and idle power consumptions of host h, and $\alpha(h)$ and $\beta(h)$ are the coefficients for low (i.e. $\leq \mathcal{L}(h)$) and high (i.e. $> \mathcal{L}(h)$) CPU load levels.

2.4.2 Network transfer power model

For both network and disk energy consumption, I assume a linear relationship between load and power consumption. However, I need to do further considerations for network modelling. First of all, in a typical data centre storage is shared on the network between all the PMs. Therefore, writing on disk necessarily results in a network transfer. For this reason, I provide a more detailed model for it, considering parameters such as (1) the packet size that I use over the network, (2) the amount of simultaneous connections and (3) the communication patterns. Plus, since in data centres different types of network interfaces are available, I build a model able to capture how these interfaces work. I distinguish two types of connections: the classical CSMA-CD connection used in technologies like Ethernet and RDMA connections, used in technologies like Infiniband. I made this choice because these two interconnection technologies are both widely used in typical data centres. I refer to the first type as datagram connections and to the latter as connected connections. In Section 4.2, I describe in detail both technologies and how differently they impact energy consumption. In the next section I describe the factors that are impacting the most the network energy consumption.

Energy-impacting factors

I describe the main factors affecting the energy consumption of network transfers according to my studies.

- *Time* this parameter must be considered since the longer a network transfer, the more energy it consumes.

- *Transport protocol* affects energy consumption because it defines the way in which transfers are performed. It defines how application layer's effective data are encapsulated. Such encapsulation inherently affects the NIC's operational mode and the amount of transferred data. While there exist many transport protocols (e.g. TCP, UDP, RSVP, SCTP), I only focus my analysis on TCP, the most pervasive one.

- *Per-packet payload size* is the real data transmitted with a single packet, juxtaposed to a header that makes the communication possible. The payload size depends on many factors such as protocol configuration, physical layer MTU, maximum segment size (MSS, representing the largest amount of data that can be sent in a single packet) on TCP, and other application characteristics (e.g. some applications require frequent exchange of small packets). Payload size has an impact on time, since a smaller payload size implies a higher number of packets and thus, more headers to process.

- *Number of connections* is the number of simultaneous connections to the NIC, typically shared among multiple applications that simultaneously send and receive data. With an increasing number of connections, one could experience a higher energy consumption due to the overhead introduced by their arbitration.

- *Traffic patterns* are different types of traffic generated by network-centric applications as showed in [97], characterised by the inter-arrival time of packets.

Power model

A network transfer occurs in two different directions: sending and receiving. Power draw of the network transfer is different if the transfer is a send or a receive. Therefore, I use different coefficients for send and receive. I model the power draw of network transfer as:

$$P^x_{net}(h,t) = \delta_x \cdot load_{net}(h,t) + K_x(c_x(t)), \tag{2.21}$$

where $x \in \{s,r\}$ and s,r refer respectively to send and receive. $load_{net}(h)$ models the network load on host h and δ_x the linear relationship between network load and power draw. Concerning the $K_x(c_x(t))$, it models the hardware related constant of the NIC, plus the overhead $O_{net}(c_x(t))$ given by the number of simultaneous connections at time t, $c_x(t)$, formally:

$$\mathcal{K}_x(c_x(t)) = \mathcal{K}_x + O_{net}(c_x(t)), \tag{2.22}$$

Regarding the overhead of multiple connections, since Gigabit and Infiniband datagram use the NICs in a different way compared to Infiniband connected, their arbitration of multiple connections is different too. For this reason, I employ Equation 2.23 for both Gigabit and Infiniband datagram and Equation 2.24 for Infiniband connected:

$$O_{\texttt{datagram}}(c_x(t)) = \log(\epsilon \cdot c_x(t) + \zeta), \tag{2.23}$$

$$O_{\texttt{connected}}(c_x(t)) = \epsilon \cdot c_x(t)^{\zeta}, \tag{2.24}$$

where ϵ and ζ model the cost in terms of power draw of arbitrating multiple simultaneous connections. When modelling network transfer, further parameters that affects the time t_{end} at which the network transfer ends need to be considered. This time is determined by (1) the network bandwidth on host h, $BW^{max}_{net}(h)$ and (2) the delay caused by transfer patterns, modelled by the parameters b_x and t_x, as in:

$$t_{end} = \frac{\texttt{DATA}_x}{\texttt{BW}^{max}_{net}(h)} + \texttt{delay}(\texttt{DATA}_x, b_x, t_x), \tag{2.25}$$

and $\texttt{delay}(\texttt{DATA}_x, b_x, t_x)$ is defined as follows:

$$\texttt{delay}(\texttt{DATA}_x, b_x, t_x) = \theta \cdot \frac{\frac{\texttt{DATA}_x}{\texttt{BW}_{net}(h)}}{b_x} \cdot \iota \cdot t_x, \tag{2.26}$$

where b_x and t_x the size of burst and throttle intervals in ms. $\texttt{DATA}_x$ is the real number of bytes transferred, that is the sum of actual data transferred by the application, plus the additional data needed for routing and transfer control, that is calculated according to

$$\texttt{DATA}_x = \texttt{PAYLOAD}_x + \frac{\texttt{PAYLOAD}_x}{p_x} \cdot \texttt{HEADER}, \tag{2.27}$$

where $\texttt{PAYLOAD}_x$ is the quantity of data to be sent/received, p_x is the payload per packet and HEADER the size of the header of each packet, whose size depends on the network protocol. Coefficients $\delta_x, \theta_x, \iota_x, \epsilon_x$ and ζ_x are calculated through linear regression. Their value is summarised in Table 4.4 together with the model error.

2.4.3 Disk power model

For power draw of disk operations, I slightly modify the model proposed by [94] by including only write operations, since in this scenario I expect to experience more writes than read, as explained in 2.5:

$$P_{\text{io}}(h,t) \quad = \quad \delta(h) \cdot P_r(h) \cdot load_{\text{io}}(h,t), \tag{2.28}$$

where $\delta(h)$ models the linear relationship between disk load and power draw coefficients.

2.4.4 VM migration model

In this section, I introduce the model for the energy consumption of each previously described migration phase. The energy consumption of the complete VM migration process is the sum of the energy consumption of each phase. I formally define VM migration as transferring the state of a migrating VM

v from a source host $\mathcal{S}$ to a target host $\mathcal{T}$. As I showed in Section 2.3.1, VM migration goes through different energy consumption phases. To delimit these phases, for each migration, I define m_s as the moment when the migration starts; t_s and t_e the time instances when the transfer phase of the migration starts and ends; and m_e as the instant when the migration ends. Thus the following time intervals define the phases: (1) between m_s and t_s is the initiation phase, (2) between t_s and t_e is the transfer phase and (3) between t_e and m_e is the activation phase, meaning, the time instance when $\mathcal{V}^*(h,t)$ changes. v represents the migrating VM, $\mathcal{S}$ the source host and $\mathcal{T}$ the target host of the VM migration. Finally, I define $\mathcal{M}(h,t)$ as the set of all the m whose $m_s \geq t$, $m_e \leq t$ and $h \in \{\mathcal{S}, \mathcal{T}\}$.

Resource utilisation model

According to the analysis in Section 2.3, the most impacting actors for VM migration are the source host ($\mathcal{S}$), the target host ($\mathcal{T}$) and the migrating VM (v). In this section I present a model for each actor's resource utilisation, to which their energy consumption is directly correlated. I define resource utilisation for VM migration $\mathcal{R}$ as a vector:

$$\mathcal{R} = [\mathcal{R}(\mathcal{S},t), \mathcal{R}(\mathcal{T},t), \mathcal{R}(v,t)], \tag{2.29}$$

reflecting the actors in Section 2.3.1. I define for both $\mathcal{S}$ and $\mathcal{T}$ hosts as well as for the migrating VM v a vector $\mathcal{R}$ where each element of the vector represents the usage of the given resource at the instant t. Both hosts and the VM have different types of resource use (e.g. CPU, memory, network, disk), as in definitions 3 and 5. However, according to the analysis in Table 3.2, the most impacting parameters on migration are: (1) CPU utilisation of the source $load_{cpu}(\mathcal{S},t)$, target $load_{cpu}(\mathcal{T},t)$, and migrating VM $load_{cpu}(v,t)$ at time instance t, (2) memory dirtying ratio $\texttt{DR}(v,t)$ of the VM v expressed in the percentage of pages marked as dirty at time instance t, (3) the amount of memory $RAM_{max}(v)$ allocated to the migrating VM v, and (4) beside these, it is also necessary to consider that migration involves transferring the state of the migrating VM over the network from source to target. Therefore, I

consider the available network bandwidth $\texttt{BW}_{net}(h,t)$ between hosts for transferring the state of the migrating VM. If the VM is idle or suspended, then $load_{cpu}(v,t) = 0$ and $\texttt{DR}(v,t) = 0$. Otherwise, I define the memory dirtying ratio $\texttt{DR}(v,t)$ as a value between 0 and 1, modelling the percentage of memory pages that are marked as dirty at the instant t.

Concerning the memory-related parameters, there are three parameters that are influencing energy consumption: memory size, memory bandwidth and memory dirtying ratio. Memory size is the amount of RAM allocated to the VM. It affects the migration time because an higher amount of memory allocated to the VM results in a longer migration time, regardless of the workload the VM is running. I consider this parameter in this energy model, because it has an impact on the migration time. Concerning memory bandwidth, it is the amount of bytes that can be read or written during a given interval of time. I do not consider it in this model because according to my analysis (see Section 2.3.1) rather then how many bytes are written in RAM, what matters is the dirtying ratio, that is the amount of memory pages that are modified while a migration is occurring. For this reason, I choose to focus only on two memory-related parameters: memory size and memory dirtying ratio. As migration is used in data centres, VMs can run on different hosts in different intervals of time. Therefore, I define for each VM a vector of triples $(h, t_{arrival}, t_{leave})$, each triple meaning that the VM v has been running on the host h in the time interval between $t_{arrival}$ and t_{leave}. In this context, $\mathcal{V}^*(h,t)$ (see Definition 3) can be seen as all the VMs in $\mathcal{V}^*(h)$ whose $t_{arrival} \leq t$ and whose $t_{leave} > t$. For this reason, resource utilisation of VM migration is defined as follows:

$$\begin{aligned} &\mathcal{R}_{\texttt{vm}}(v,t) = \{load_{cpu}(v,t), \texttt{RAM}_{max}(v), \texttt{DR}(v,t)\}, \\ &\mathcal{R}(h,t) = \{load^{\texttt{virt}}_{cpu}(h,t), \textstyle\sum_{v \in \mathcal{V}^*(h,t)} \mathcal{R}_{\texttt{vm}}(v,t), \texttt{BW}_{net}(h,t)\}, \end{aligned} \tag{2.30}$$

where $load_{cpu}(v,t)$ is the CPU usage of the VM v at the instant t, that is dependent from the workload the VM is running. $\texttt{RAM}_{max}(v)$ represents the memory size allocated to VM v, defined as number of memory pages. $\texttt{DR}(v,t)$ is the dirtying ratio of the workload running on the VM at the time

t. Concerning $\mathcal{R}$, $\texttt{BW}_{net}(h,t)$ is the bandwidth available on the host h at the instant t. The parameters $load_{cpu}(\mathcal{S},t)$ and $load_{cpu}(\mathcal{T},t)$ mainly depend on three terms:

1. CPU utilisation $load^{vmm}_{cpu}(h,t)$ for arbitrating the hardware resources shared among the VMs,

2. CPU utilisation $load_{cpu}(v,t)$ of each VM v executed on the host h at the instant t,

3. CPU load $load^{migr}_{cpu}(h,t)$ added by migration on both source and target.

Therefore, I detail the previous definition as follows:

$$load_{cpu}(h,t) = load^{vmm}_{cpu}(\mathcal{V}^*(h,t)) + \sum_{v \in \mathcal{V}^*(h,t)} load_{cpu}(v,t) + load^{migr}_{cpu}(h,t), \tag{2.31}$$

where $\mathcal{V}^*(h,t)$ is the complete set of VMs running on the host $h \in \{\mathcal{S},\mathcal{T}\}$ at time instance t other than the migrating VM v. Equation 2.31 formalizes the total CPU utilisation for host h at instant t:

$$load^{virt}_{cpu}(h,t) = load^{vmm}_{cpu}(\mathcal{V}^*(h,t)) + \sum_{m \in \mathcal{M}(h,t)} load^{migr}_{cpu}(m,t). \tag{2.32}$$

Finally, I define the CPU usage of the whole PM as follows:

$$load_{cpu}(h,t) = load^{virt}_{cpu}(h,t) + \sum_{v \in \mathcal{V}(h,t)} load_{cpu}(v,t). \tag{2.33}$$

In Equation 2.31 I do not consider how arbitration of CPU resources between the Virtual Machine Monitor (VMM) and migrating VM is performed, because I observed that it does not affect power consumption.

Energy model

First, I define the consumption of a VM v running in a PM h, which I denote as $E_{vm}(v,h)$. Power consumption of a VM v on a host h depends on two main parameters: (1) resource usage of v and (2) the time interval ($t_{arrival}$,

t_{leave}) in which the VM is running on host h. Moreover, the power drawn of v also depends on the host that is running v. Therefore, the host that is executing the VM needs to be included as parameter in order to use its instantaneous in the definition. For simplicity, I assume that if a VM is not running or has been suspended, its dynamical power on the host h is equal to 0. Therefore, I define the consumption of the VM as $E_{vm}(v, (t_{arrival}, t_{leave}))$, as shown in the following equation:

$$E_{vm}(v, t_{arrival}, t_{leave}) = \int_{t_{arrival}}^{t_{leave}} P(\mathcal{R}_{\texttt{vm}}(v, t))\ dt. \tag{2.34}$$

Next, as I consider also energy consumption given by virtualization, I extend the equation of the active energy consumption of the host h (see Equation 2.14) as follows:

$$E_{active}(h) = \int_{t_{start}}^{t_{end}} P_{active}(h) + P_{virt}(h, t)\ dt. \tag{2.35}$$

Concerning P_{virt}, this power draw is given by (1) the arbitration of the resource requirements of the VMs running on host h and (2) the power consumption of the VMs management tasks (e.g, migration, shutdown, startup). For this model, I focus only on the VM migration energy consumption. The first is dependent on the resource usage of the host and on the resource demand of each VM executed on the host h, the latter on migration-related parameters. To describe the power consumption of virtualization, I finally define $P_{virt}(h, t)$:

$$P_{virt}(h, t) = P_{\texttt{vmm}}(\mathcal{R}(h, t)) + \sum_{m \in \mathcal{M}(h,t)} P_{\texttt{migr}}(m, h, t), \tag{2.36}$$

where $P_{\texttt{vmm}}$ is the power consumed by the VMM to handle the contention of resources by the VMs in the host, while $P_{\texttt{migr}}$ the power consumed by migration, that is equal to 0 when $\mathcal{M}(h, t) = \emptyset$. In the following I do not consider $P_{\texttt{vmm}}$, as I do not expect it to vary significantly over the time. Therefore, I focus on the power consumption of VM migration, $P_{\texttt{migr}}$. As I explain in Section 2.3.1, the VM migration power draw varies according to the energy

phase that the VM migration is performing. Therefore, I define $P_{\texttt{migr}}$ as the sum of the power draw in each phase:

$$P_{\texttt{migr}}(h, v) = P^{(\texttt{i})}(h, v, t) + P^{(\texttt{t})}(h, v, t) + P^{(\texttt{a})}(h, v, t). \tag{2.37}$$

Then, I consequently define energy consumption of VM migration as follows: for each PM $h \in \{\mathcal{S}, \mathcal{T}\}$, the energy consumption of the migration is the integral of the instantaneous power drawn caused by the migration process throughout its duration $[m_s, m_e]$:

$$E_{\texttt{migr}}(h, v) = \int_{m_s}^{m_e} P_{migr}(h, v)\ dt = \int_{m_s}^{m_e} P^{(\texttt{i})}(h, v, t) + P^{(\texttt{t})}(h, v, t) + P^{(\texttt{a})}(h, v, t)\ dt, \tag{2.38}$$

where the power drawn is represented as the sum of the power consumed over the three phases identified in Section 2.3.1: initiation $P_{(\texttt{i})}$, transfer $P_{(\texttt{t})}$, and activation $P_{(\texttt{a})}$. The following subsections discuss the model for each of these power drawn functions. Integrating these values over the migration time, I obtain the energy consumption over each phase, $E_i(h, v)$, $E_t(h, v)$ and $E_a(h, v)$, respectively. After defining power consumption for each VM migration phase, I move on defining energy consumption of the whole VM migration process. First, I define a migration m as a quadruple $((m_{start}, t_{start}, t_{end}, m_{end}), v, \mathcal{S}, \mathcal{T})$, where m_{start} is the instant in which the migration starts, t_{start} and t_{end} are the instants in which, respectively, the transfer phase of the migration starts and ends, and m_{end} is the instant in which the migration ends. The time interval between m_{start} and t_{start} is the one during which initiation phase takes place and the one between t_{end} and m_{end} is when activation phase takes place. v represents the migrating VM and $\mathcal{S}, \mathcal{T}$ represent respectively source and target hosts. I define energy consumption of each phase, where $E_i(h, v)$ is the energy consumed in the initiation phase:

$$E_i(h, v) = \int_{m_s}^{t_s} P^{(\texttt{i})}(h, v, t)\, dt, \tag{2.39}$$

$E_t(h, v)$ is the energy consumed over by transfer phase:

$$E_t(h, v) = \int_{t_s}^{t_e} P^{(\mathtt{t})}(h, v, t)\, dt, \tag{2.40}$$

and $E_a(h, v)$ is the energy consumed by the activation phase:

$$E_a(h, v) = \int_{t_e}^{m_e} P^{(\mathtt{a})}(h, v, t)\, dt, \tag{2.41}$$

where the power functions are subject to the following constraints:

$$\begin{aligned} &P^{(i)}(\mathcal{R}(h,t), \mathcal{R}_{\mathtt{vm}}(v,t), m) = 0 \text{ if } t > t_s \text{ or } t < m_s, \\ &P^{(t)}(\mathcal{R}(h,t), \mathcal{R}_{\mathtt{vm}}(v,t), m) = 0 \text{ if } t < t_s \text{ or } t > t_e, \\ &P^{(a)}(\mathcal{R}(h,t), \mathcal{R}_{\mathtt{vm}}(v,t), m) = 0 \text{ if } t < t_e \text{ or } t > m_e. \end{aligned} \tag{2.42}$$

In the following, I differentiate the power functions according to the host I model. According to the analysis I performed in Section 2.3.1, the energy consumption of each phase is influenced by different actors. I give more detail about this in the following sections, in which I describe the power draw functions during each VM migration phase. To improve readability, I simplify the notation of each function as follows:

$$\begin{aligned} P^{(i)}(\mathcal{R}(h,t), \mathcal{R}_{\mathtt{vm}}(v,t), m) &= P^{(i)}(h, v, t), \\ P^{(t)}(\mathcal{R}(h,t), \mathcal{R}_{\mathtt{vm}}(v,t), m) &= P^{(t)}(h, v, t), \\ P^{(a)}(\mathcal{R}(h,t), \mathcal{R}_{\mathtt{vm}}(v,t), m) &= P^{(a)}(h, v, t). \end{aligned} \tag{2.43}$$

In the next subsections, I provide more details about each phase's power consumption.

Initiation phase

In this phase, I expect the power consumption on both hosts to depend on: (1) the increase in CPU usage for initiating VM migration and (2) the additional CPU usage for suspending the VM on the source host. On the source host, I also consider the resource usage of the VM that is be running over this phase: Since I observed that power consumption is proportional to the CPU load of the host and of the VM, I employ Equation 2.44 to estimate

power consumption on both hosts:

$$P^{(i)}(h,v,t) = \alpha^m_{(\mathtt{i})}(h) \cdot \mathcal{R}(h,t) + \beta^m_{(\mathtt{i})}(h) \cdot \mathcal{R}_{vm}(v,t) + \mathcal{C}^m_{(\mathtt{i})}(h)$$
$$= \alpha^m_{(\mathtt{i})}(h) \cdot load_{cpu}(h,t) + \beta^m_{(\mathtt{i})}(h) \cdot load_{cpu}(v,t) + \mathcal{C}^m_{(\mathtt{i})}(h), \quad (2.44)$$

where $\alpha^m_{(\mathtt{i})}(h)$ and $\beta^m_{(\mathtt{i})}(h)$ model the relationship between the CPU usage of the two PMs and of the migrating VM to the power consumption, and $\mathcal{C}^m_{(\mathtt{i})}(h)$ include the power consumption the for establishing a connection between the two PMs. I approximate the power consumption with a linear function, as done in [98]. On the source host, it also includes the power consumption for suspending the VM. As the target is not yet involved in the execution of the VM, in the equation for $\mathcal{T}$ I set $load_{cpu}(v,t) = 0$.

Transfer phase

Since transferring the state of the VM from the source to the target host is a network-intensive process, its power consumption is mainly related to the network bandwidth. In this phase, I also consider the CPU usage on both hosts proportional to the power consumption, while I also expect a linear relationship between dirtying ratio and power consumption due to the increased contention on memory.

$$P^{(t)}(h,v,t) = [\alpha^m_{(\mathtt{t})}(x), \beta^m_{(\mathtt{t})}(x)]' \cdot \mathcal{R}(h,t) + [\gamma^m_{(\mathtt{t})}(x), \delta^m_{(\mathtt{t})}(x)]' \cdot \mathcal{R}_{vm}(v,t) + \mathcal{C}^m_{(\mathtt{t})}(x)$$
$$= \alpha^m_{(\mathtt{t})}(h) \cdot load_{cpu}(h,t) + \beta^m_{(\mathtt{t})}(h) \cdot \mathtt{BW}_{net}(h,t) + \gamma^m_{(\mathtt{t})}(h)$$
$$\cdot \mathtt{DR}(v,t) + \delta^m_{(\mathtt{t})}(h) \cdot load_{cpu}(v,t) + \mathcal{C}^m_{(\mathtt{t})}(h), \quad (2.45)$$

where $\alpha^m_{(\mathtt{t})}(h)$ models the linear relationship between power and CPU usage, $\beta^m_{(\mathtt{t})}(h)$ the relationship between bandwidth and power, $\gamma^m_{(\mathtt{t})}(h)$ the linear relationship between the dirtying ratio and power consumption, $\delta^m_{(\mathtt{t})}(h)$ the linear relationship between the migrating VM's CPU usage and its power consumption and $\mathcal{C}^m_{\mathtt{t}}(h)$ the power consumption for moving the state of the migrating VM to the target host. I expect the latter to be higher on the target host than on the source because it also needs to load the VM state

in memory. The main difference between live and non-live migration is that during a live migration, the migrating VM is still running on the source host and, therefore, I consider the power consumption on the host due to its workload (i.e., $\texttt{DR}(v,t) \neq 0$ and $load_{cpu}(v,t) \neq 0$). As the VM is not yet on the target, both the dirtying ratio and the migrating VM's CPU utilisation becomes 0 while evaluating power consumption on the target host.

In this case I also consider the CPU usage of the migrating VM, since it is still running during the transfer phase.

Activation phase

After the transfer phase is completed, there are two remaining actions to be performed: resuming the VM on the target host and deallocating the resources occupied on the source host. Afterwards, due to the release of the resources previously owned by the migrating VM, on the source host, I consider the CPU load and a constant power consumption $\mathcal{C}^m_{(\texttt{a})}(\texttt{S})$ only. Concerning the target host, I consider the power consumed by the migrating VM that starts its execution, as well as the constant power consumed by the hypervisor to start the VM $\mathcal{C}^m_{(\texttt{a})}(\texttt{T})$:

$$\begin{aligned} P^{(a)}(h,v,t) &= \alpha^m_{(\texttt{a})}(h) \cdot \mathcal{R}(h,t) + \beta^m_{(\texttt{a})}(h) \cdot \mathcal{R}_{vm}(v,t) + \mathcal{C}^m_{(\texttt{a})}(h) \\ &= \alpha^m_{(\texttt{a})}(h) \cdot load_{cpu}(h,t) + \beta^m_{(\texttt{a})}(h) \cdot load_{cpu}(v,t) + \mathcal{C}^m_{(\texttt{a})}(h), \quad (2.46) \end{aligned}$$

where $\alpha^m_{(\texttt{a})}(h)$ models the linear relationship between CPU usage and power consumption, and $\beta^m_{(\texttt{a})}(h)$ models the relationship between the CPU usage of the starting VM.

2.5 Runtime modelling

In this work, I aim to simulate not only power consumption but also the VM migration time. VM migration is the process of transferring the VM state from source to target. I distinguish between two types of VM migration: non-live and live migration. In the non-live migration, the VM state is transferred

after suspending the VM on the source and then resuming it on the target host. In the live migration, the state of the VM is transferred while the VM is still running. For both migration types, I identified in Section 2.3.1 three VM migration phases:

- *Initiation phase*, during which source host prepares transferring the VM state to the target host and the target reserves the resources necessary to host the VM;
- *Transfer phase*, during which the VM state is transferred from the source to the target host in a way depending on whether a non-live or live migration is performed;
- *Activation phase*, during which the source host frees the resources occupied by the VM and the target starts it.

I therefore define the VM migration time $T_{\text{migr}}(v, h, \mathcal{S}, \mathcal{T})$ on host h for migrating the VM v from the source $\mathcal{S}$ to the target host $\mathcal{T}$ as the sum of the times required in each phase:

$$T_{\text{migr}}(v, h, \mathcal{S}, \mathcal{T}) = T_{init}(v, h) + T_{\text{transf}}(v, h, \mathcal{S}, \mathcal{T}) + T_{\text{activ}}(v, h). \tag{2.47}$$

In the initiation phase, the source host prepares a checkpoint of the VM to be sent to the target. In the activation phase, the source host frees the resource allocated to the VM and the target starts it. Therefore, the times required by both initiation $T_{init}(v, h)$ and activation $T_{activ}(v, h)$ phases are only dependent on the VM size $\texttt{RAM}_{max}(v)$ and the storage bandwidth on the host h:

$$T_{\text{init}}(v, h) = T_{\text{activ}}(v, h) = \frac{\texttt{RAM}_{max}(v)}{\texttt{BW}_{\text{io}}(h, t)}. \tag{2.48}$$

The transfer phase, on the other hand, has a different execution time for a live or a non-live migration. The non-live migration time $T^{\text{nonlive}}_{\text{transf}}$ depends only on the VM size and the bandwidth on the host h at instant t, $\texttt{BW}_{net}(h, t)$:

$$T^{\text{nonlive}}_{\text{transf}} = \frac{\texttt{RAM}_{max}(v)}{\texttt{BW}_{net}(h, t)}. \tag{2.49}$$

Live migration is instead performed iteratively while the VM v is still running. Therefore, the VM state needs to be continuously updated over a predefined number of iterations, set in the hypervisor's configuration. After the initial state transfer, during each iteration only the memory pages that have been modified during the previous transfer of the VM state are transferred, leading to the following live VM transfer time:

$$T^{\text{live}}_{transf} = \frac{\texttt{RAM}_{max}(v)}{\texttt{BW}_{net}(h,t)} + \sum_{i=1}^{\mathcal{I}} \frac{\texttt{DP}(v,i)}{\texttt{BW}_{net}(h,t)}, \tag{2.50}$$

where $\mathcal{I}$ is the number of iterations and:

$$\texttt{DP}(v,i) = \frac{\texttt{RAM}(v)}{\texttt{PS}(v)} \cdot \texttt{DR}(v,i), \tag{2.51}$$

where $\texttt{DR}(v,i)$ is the dirtying rate of the VM v or the percentage of memory pages marked as dirty during an iteration i, and $\texttt{PS}(v)$ is the size of each memory page of VM v.

Section 6.3 gives implementation details of this model.

2.6 Summary

In this chapter, I formally defined all the actors that are involved in this work. Then, since the main focus of this work is the modelling of VM migration, I focused on defining a VM and its role inside a data centre. Afterwards, I defined the VM migration as the activity that allows the data centre management to perform VM consolidation. Then, I gave deep insights on VM migration, describing the difference between non-live and live migration and its power characteristics. Then, I proposed a model for data centre energy consumption, considering all the actors involved in VM migration. Afterwards, I defined VM migration as a network intensive process, by developing a simple network transfer energy consumption model that I further use to predict energy consumption of VM migration. Then, I designed an extended version of the model, taking into account not only network transfer consumption but also VM workload and all the actors that are involved in the VM

migration process. In the next chapters I validate and implement this model, showing its improved accuracy compared to the state-of-the-art.

Chapter 3

Experimental methodology

3.1 Introduction

In this chapter I introduce the experimental methodology that has been used to validate my work. First, I describe the different possibilities to measure energy consumption of the PMs, explaining the motivations behind the choice of the selected measurement framework. Afterwards, I describe more in detail the measurement framework, and the additional hardware I employed to measure energy consumption of the PMs that were the targets of my experiments. Afterwards, I describe the benchmarks I use to generate the type of load that is needed for my experiments and to get measurements that are needed to generate the training and test set for the model's validation, respectively for network transfers and for VM migration.

3.2 Motivation

In most of the existing work about energy prediction in data centres, like [33, 32, 96], authors consider only energy consumption generated by CPU. However, to provide a more accurate estimation, my model considers also network and storage energy consumption, as modelled respectively in Equations 2.16 and 2.17. Therefore, for validation purposes, I needed to get power measurements not only related to CPU load but also to each one of these components.

To get these measurements, I needed fine grain measurements for each PM component. Concerning CPU, technologies like RAPL [99] are available on some architectures for measuring power draw, but such components are still not available for other components. For these reasons, I resorted to use external power measurement devices. However, such power measurement devices allow to measure only the power consumption of the whole machine, making difficult to isolate power draw that is generated by a specific component. Therefore, I needed a way to isolate power draw for each component. To this end, I decided to use an approach that allows me to measure the machine power draw through the external power measurement devices, but only during the execution of specific portion of application source code. Such an approach is called code instrumentation. For example, if the measurements are taken during the execution of a CPU intensive portion of code, I can assume that the power draw I measure in this phase is stemming mostly from the CPU load. Conversely, if I measure during the execution of a network transfer, I can assume that the measured power draw is related to the network transfer. After defining the measurement framework, I needed also applications capable of loading specific PM's components, so that I can extract data about the power draw of each one of the selected components. For example, if I want to measure energy consumption of the network, I need applications that are loading mostly the network, such as intensive network transfers over TCP sockets and so on. From this applications, I need to isolate the portions of code that are specific to the network transfers and instrument them using the selected framework. After instrumenting them, I measured power draw of these portions of code. I give more details about it in the next sections.

3.3 Code instrumentation framework

In this section I describe the way I get the power measurements I need for my model validation. An example of the way such framework works is given in Figure 3.2, where the highlighted code lines describe the way a measurement is performed and how the data is collected. Of course there will be some resource arbitration and scheduling related power draw involved, but for sure

less than measuring the power consumption over the whole application, giving a precise way to estimate energy consumption of different components. This framework exploits the API offered by the power measurement device that I used during this work, the `Voltech PM1000+`[1]. The framework relies on a third part, called the PM server, that is responsible for managing the power measurement devices and gathering the measurements from them. The operation mode is summarised by Figure 3.1. The power measurement devices are attached to the PMs. The application is compiled by including the power measurement library and by adding the functions that will determine the start and the end of the measurements. Once the instrumented application reaches a code region whose power draw I measure, the measurement collecting phase goes through the following steps:

1. *pmStartSession()*: a message from the PM to the PM server is sent, instructing it to start the power measurements;

2. *connection with devices*: once the PM server receives the request of starting a measurement session, it will instruct the power measurement devices to start measuring the power draw on the PM of interest;

3. *collect measurements*: during this phase, the PM server reads the the power draw of the PM through power measurement devices, storing them in its hard drive;

4. *pmStopSession()*: a connection from the PM to the PM server is performed, to tell it to stop collecting measurements. Once the PM server receives this message, interrupts the connection with the devices and saves the data collected until now;

5. *measurements retrieval*: finally, the power measurements device has read until now are saved on the PM, ready to be further analysed.

It is important to note that once the measurement session starts, no further interaction happens with the PM server until the instrumented binary tells it to stop its measurements. I do this in order to reduce overhead on

[1] `http://www.farnell.com/datasheets/320316.pdf`

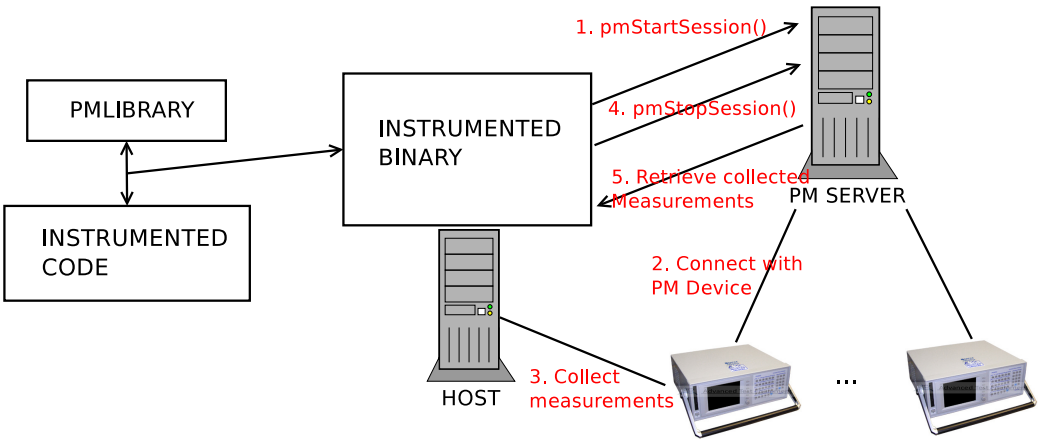


Figure 3.1: Code instrumentation framework.

```
self.eneI.startSession(taskId)
for i in range(len(A)):
    for j in range(len(B[0])):
        for k in range(len(B)):
            C[i][j] += A[i][k]*B[k][j]
self.eneI.stopSession()
energyReadings = self.eneI.getEnergy()
```

Figure 3.2: Example of usage of the code instrumentation framework.

the network and to add to the PM some computation/communication cost that could affect the accuracy of my measurements. In the next section, I describe the applications that I use to collect the measurements I needed for validating my model.

3.4 Network benchmarking

In this section I describe the benchmarking methodology for evaluating the energy consumption of the NIC software stacks.

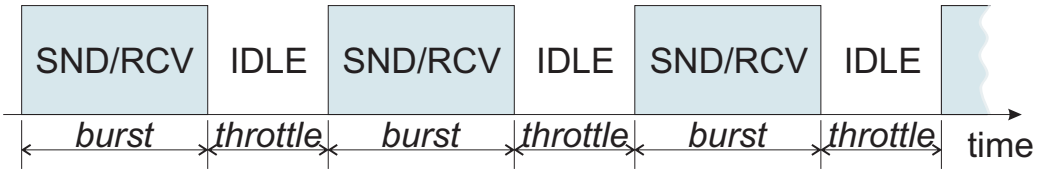


Figure 3.3: PATTERN benchmark (burst/throttle intervals).

3.4.1 Experimental design

Network benchmarks

I investigate each factor through six benchmarks. Each benchmark is studied to test the impact of different network energy impacting factors, that I outlined in Section 2.4.2. All the benchmarks run on TCP transport protocol.

BASE benchmark investigates the impact of the network transfer on the energy consumption by transferring a fixed amount of data using sockets without any specific tuning.

PSIZE benchmarks investigate whether the NIC energy consumption is related to the payload size under two premises: (1) *PSIZE-DATA* determines the impact of the payload size on energy efficiency independent of the data size by repeatedly transferring a fixed amount of data while varying the maximum payload size, and (2) *PSIZE-TIME* performs a maximum payload size evaluation with a fixed transfer time by continuously transferring data until the timeout I set is reached.

n-UPLEX benchmark evaluates the energy consumption of NICs in full duplex (FD) mode, while handling multiple concurrent connections. I transfer a fixed amount of data using a varying number of FD connections on each machine.

PATTERN benchmarks evaluate the effects of traffic patterns on energy consumption. I transfer data multiple times, and configure the data transmissions to be a succession of *burst* and *throttle* intervals, representing fixed

Tool	*Transfer data size*	*Transfer timeout*	*MSS setting*	*Disable buffering*[1]	*FD/HD connections*	*Concurrent connections*	*Variable pattern*
`ttcp`[2]	✓	✗	✗	✓	✗	✗	✗
`netperf`[3]	✓	✓	✗	✓	✗	✗	✗
`iperf` [4]	✓	✓	✓	✓	✓	✓	✗

[1] e.g. an option for setting the TCP_NODELAY
[2] version 1.12
[3] version 2.4
[4] version 2.05

Table 3.1: Comparison of networking benchmarking/diagnosis tools.

time intervals in which the NICs are continuously communicating and idle, as depicted in Figure 3.3. For *PATTERN-B* I keep the throttle size constant and vary the burst size, while *PATTERN-T* I vary the throttle size keeping a constant burst size.

For the PSIZE benchmarks, I need to successively set the transferred data size and a transmission timeout, and to strictly control the packet size. This can be achieved by altering the MSS and by disabling any buffering algorithms. For the n-UPLEX benchmark, I need to configure the type of (FD/HD) connections and the number of simultaneous connections. Finally, the PATTERN benchmark requires the possibility to shape the communication patterns through variable burst and throttle intervals. In the next section, I describe how I implemented these benchmarks.

Network benchmarks implementation)

To configure the metrics of my study based on transfer data size and timeout, payload size, FD/half-duplex (HD) connections, connection concurrency, and transmission patterns, I analyzed three of the most popular open-source network diagnosis and benchmarking tools: `ttcp`[2], `netperf`[3] and `iperf`[4]. Table 3.1 presents a comparison of the flexibility of these tools focused on the provided configuration options for the metrics relevant to

[2] `http://www.pcausa.com/Utilities/pcattcp.htm`
[3] `http://www.netperf.org/netperf/`
[4] `http://iperf.sourceforge.net/`

this study. Since none of the analysed tools covers all configuration parameters needed, I designed the *Nimble NEtwork Traffic Shaper (NNETS)*, a versatile network traffic shaping tool implemented in Python 2.7 using the standard socket API, publicly available under GNU GPL v3 license at the URL `https://github.com/vincenzo-uibk/nnets`). In addition to the custom design required for accommodating all studied configurations, the tool allows a proper instrumentation of network and energy metrics. I implemented it with a clear separation between data processing and networking operations in order to instrument only the relevant regions of code, excluding data staging and pre-/post- processing operations and ensuring that the measured energy consumption is strictly related to the network transfer.

To evaluate software stacks' energy efficiency I employ the following five metrics:

- *Machine energy consumption* defined in Equation 2.35, in Kilojoules kJ, defining the energy consumption of the whole PM subsystems for running each experiment;

- *Network energy consumption* defined in Equation 2.16, in Kilojoules kJ,computed as the difference between the machine's energy consumption during benchmarks' execution and its idle consumption. This metric includes the energy consumed by all hardware components involved in a network transfer, which I purposely include to have a more realistic metric related to the software application and not to the hardware;

To have further indications about energy and power behavior of each NIC, I also compute the following metrics:

- *Average power* in Watts (W), defined as the ratio between network energy consumption and its execution time;

- *Energy per byte* in Nanojoules nJ,defined as the ratio between the network energy consumption and the number of bytes transferred, which indicates how energy consumption varies in relation to the size of data transfer;

- *Energy per packet* in Millijoules mJ, defined as the ratio between the network energy consumption and the amount of packets transferred.

3.5 VM migration benchmarking

3.5.1 Experimental design

In this section, I introduce the methodology to evaluate the VM migration model's accuracy: first, I describe the rationale behind the experimental design, then I introduce the hardware and software configuration for my energy measurements.

Workloads

The three selected actors can influence the energy consumption of VM migration in different ways, especially depending on the workloads they are running. I analyse this aspect in the following paragraphs. Each actor has a different influence on VM migration according to the type of workload that is currently running.

Although there may be different kind of workloads running in a data centre (e.g. CPU-intensive, memory-intensive, network-intensive, or mixed), in the following, I focus on the CPU and memory-intensive ones because they impact the VM migration process the most. CPU-intensive workloads have the highest impact on migration which is highly prone to performance degradations if one actor needs to perform migration-related computations while running a CPU-intensive workload. Concerning memory-intensive workloads that continuously update RAM locations, several transfers are needed to achieve a consistent state between the source and the target if such updates happen before the state is transferred over the network. Table 3.2 summarises the workloads' impact on VM migration. When the migrating VM is running a CPU-intensive workload, a performance drop may be experienced if the source and/or target hosts are fully loaded because the host's CPU must be shared between the workload of the hosts and the newly initiated migration process. If the migrating VM is running a memory-intensive

workload that continuously updates RAM locations, it will highly impact the performance of the live migration approach since several state updates are needed to achieve a consistent VM state between the source and the target hosts. I do not consider the case when a non-live migration approach is used, because in this approach the VM will be suspended before migrating it to the target. For these reasons, I only consider in this work (1) CPU intensive workloads running on source, target and migrating VM, and (2) memory-intensive workloads running on the migrating VM. I consider as memory-intensive workloads: (1) workloads using at least 90% of the memory allocated to the VM and (2) workloads with a high memory dirty ratio (i.e. a high percentage of memory pages marked as dirty over a given amount of time).

When migrating VM is running a CPU-intensive workload, performance drop in VM migration may be experienced if either source or target are fully loaded. This is because both migrating VM and the workload of the source/target are using all the machine's CPUs. Thus, once a migration is issued it will share the CPU with the workload running on the machine and on the migrating VM. In case the source host is loaded, such an effect on migration performance will be visible when the state is transferred or even when the machine is started up, if the target is loaded. This performance drop will be visible also in the case of non-live migration. I do not consider the case where either source or target are not fully loaded, because in this case no contention is expected. If the migrating VM is running a Memory-intensive workload instead,this will impact performance of migration especially if a live migration approach is used. Such performance drops will be experienced no matter what kind of workload source or target are running and regardless of the load they are experiencing in the meanwhile. I do not consider the case when a non-live migration approach is used, because in this approach the VM will be suspended before migrating it to the target. For this reason, the state will be transferred to the target just once, with no impact on VM migration performance.

- *CPU-intensive*: If VM is running a CPU-intensive workload, perfor-

Workload	*Migration type*	*Migrating VM*	*Source host*	*Target host*
CPU intensive	LIVE	Source/target load-dependant	Slowdown in transfer	Slowdown for VM start or transfer
	NON-LIVE			
MEMORY intensive	LIVE	Multiple transfers of VM state	Slight performance degradation	Slight performance degradation
	NON-LIVE	No influence		

Table 3.2: Workload impact on VM migration according to the hosting actor.

mance drop in VM migration may be experienced if either source or target are fully loaded. This is because both migrating VM and the workload of the source/target are using all the machine's CPUs. Thus, once a migration is issued it will fight for the CPU with the workload running on the machine and on the migrating VM. Such an effect on migration performance will be visible when the state is transferred, in the case the source is loaded, or even when the machine is started up, if the target is loaded. This performance drop will be visible also in the case of non-live migration. I do not consider the case where either source or target are not fully loaded, because in this case no contention is expected.

- *Memory-intensive*: When a VM is running a memory-intensive workload, this will impact performance of migration, especially if a live migration approach is used. Such performance drops will be experienced no matter what kind of workload source or target are running and regardless of the load they are experiencing in the meanwhile. Such a performance drop will be proportional to the amount of modifications performed in the memory while the state is transferred over the network. I do not consider the case when a non-live migration approach is used, because in this approach the VM will be suspended before migrating it to the target. For this reason, the state will be transferred to the target just once, with no impact on VM migration performance.

Therefore, I need to model VM migration consumption taking into account each one of the selected workload and actors.

My experimental settings are summarised in Table 3.5a, and the VM and hardware configurations in Tables 3.5b and 3.3. I used the Xen hypervisor version 4.2.5, including both `xm` and `xl` toolstacks configured to perform the live and non-live migrations between two PMs as specified in Table 3.3. The two machines were connected through a networking switch. For my experiments, I use two hosts capable of running Xen virtualization software and configured to run a VM migration. Another configuration requirement for their is that I should be able to measure the energy consumption of the migration, excluding the impact of load that is not generated by my experiments. Moreover, I use machines capable to run a version of Xen that allows to perform both live and non-live migration. I performed the experiments on two sets of machines (`m01-m02` and `o1-o2`) with different CPUs and network cards/switch, to allow the validation of this model on different hardware configurations. For each experiment, I employed paravirtualized VMs mostly encountered in modern data centres as they ensure near-native performance. For the migrating VMs, I chose 4 GBs of memory size to assure a long enough migration time for the clear identification of the energy consumption phases.

According to the analysis in Table 3.2, CPU-intensive workloads running on source/target hosts and memory-intensive workloads running on the migrating VM have the highest impacts on the energy consumption VM migration. Therefore, I designed two families of experiments: *CPULOAD* and *MEMLOAD*.

CPULOAD

I investigate the impact of VM workload on live and non-live migration using two types of experiments: During the execution of each experiment in this family, the migrating VM is also running a CPU-intensive workload. I made this choice for clarity of presentation of my results and to clearly identify in the traces the time intervals in which the migrating VM is running. For this category of experiment I use both live and non-live migration, because according to my analysis in both cases a difference in the energy consumption

will be noticed. Two experiments belong to this category, both of them are detailed below:

CPULOAD-SOURCE investigates the impact of CPU-intensive workloads running on the source host by migrating a VM to an idle target host. The load of the source is progressively increased from idle to 100% CPU utilisation to quantify its impact on VM migration. I also consider the case in which the VMs require more CPUs than the host can offer, to ensure some multiplexing amongst them.

CPULOAD-TARGET investigates the impact of CPU-intensive workloads running on the target host by migrating a VM from a source host running the migrating VM only. The load of the target is progressively increased from idle to 100% CPU utilisation to quantify its impact. Also in this experiment, I consider the effects of multiplexing on hardware resources.

For the CPU-intensive workload, I use an OpenMP C implementation of a matrix multiplication algorithm for two reasons: it is used by many scientific workloads running on data centres, and it can be easily parallelised allowing to load all virtual CPUs of the VMs taking part in the experiments with while it introduces only small communication and synchronisation overheads. Concerning the VM configuration, I select the `load-cpu` and `migrating-cpu` type among the instances described in Table 3.5b. I employ the `load-cpu` VM instance to load the PM while migrating an instance of `migrating-cpu` type. I assign as many CPUs to these instances as needed to increase the load by 25% increments.

MEMLOAD

In this family of experiments study the effect of varying dirtying ratio in the migrating VM on the migration process. To compare the impact of the memory-intensive workloads with the CPU-intensive ones, I designed experiments involving CPU-intensive workloads running on both source and target, as follows:

MEMLOAD-VM studies the impact of memory-intensive workloads by increasing the percentage of memory pages dirtied in the migrating VM. The source host is only running the migrating VM and the target is idle. This experiment serves as the baseline for the rest of the memory intensive ones.

MEMLOAD-SOURCE investigates how live migration is differently impacted by: (1) CPU-intensive workloads running on the source host and (2) memory-intensive workloads running on the migrating VM. I perform a live migration of a VM running a memory-intensive workload from a source host running a CPU-intensive workload with increasing utilisation to an idle target.

MEMLOAD-TARGET investigates how live migration is differently impacted by: (1) CPU-intensive workloads running on the target host and (2) memory-intensive workloads running on the migrating VM. I perform a live migration of a VM running a memory-intensive workload to a target host running a CPU-intensive workload with increasing utilisation. The source host is running the migrating VM only.

These experiments employ live migrations only, since non-live migrations have $\texttt{DR}(v,t) = 0$. For this category of experiments, I chose a memory-intensive workload called *pagedirtier* implemented in ANSI C that continuously writes in memory pages in random order. I fixed the memory allocated to this application to 3.8 GB and only write in the memory pages in this part of the memory. I made this choice to avoid swapping effects incurring additional VM migration overheads, due to the continuous writing to the NFS storage and a consequent reduction of the available bandwidth. I employ again the `load-cpu` VM instances for generating load on the hosts and `migrating-mem` as the migrating VM (see Table 3.5b).

Workloads implementation

Concerning the VMs workloads, I needed CPU and Memory-intensive workloads. I developed my own implementation of such workloads, in order to (1) reduce unnecessary computation who impact on memory, for what concern

Host	*CPU*	*Kernel*	*Gigabit NIC*	*Infiniband NIC*	*dom0 kernel*	*Xen version*
`k12`	4×	Linux	Broadcom	SDR Mellanox	3.0.4	4.2
`k13`	Opteron 880	2.6.9-67	BCM5704	MT23108		

Table 3.3: Experimental hardware.

CPU-intensive and (2) specifically tune the workloads to the machines I use. The implementation is described below:

- *matrixmult*: An OpenMP C implementation of a matrix multiplication. This application represents a CPU-intensive workload. I select such workload for two main reasons: first of all, it is very common in many scientific workloads running on data centres. Secondly, it can be easily parallelized using OpenMP, thus allowing to load all the virtual CPU of the VMs with a small communication and synchronization overhead. The memory size I use is fixed to 205MB, to ensure that all the computational load of the application is CPU-bound.

- *pagedirtier*: This application represents a Memory-intensive workload. It is implemented as an ANSI C program that continuosly writes in memory pages, accessing them in random order. I fix the memory allocated to this application to 3.8GB and write only in the memory pages in this part of the memory. I made this choice to avoid the effect of swapping on the applications, as they add to the VM migration an overhead related to the continuous writing to the NFS storage.

3.6 Hardware/software configuration

3.6.1 Network benchmarks experimental setup

I employ two machines, both equipped with Infiniband and Gigabit Ethernet NICs, as specified in Table 3.3. I set the MTU on all machines to 16382 bytes for the Infiniband NICs in connected mode, to 2044 bytes in datagram mode, and to 1500 bytes for the Gigabit Ethernet NICs. The machines are

connected through two dedicated server-grade network switches to exclude the impact of external network traffic. For each NIC and connectivity mode, I run the benchmarks in three configurations (send, receive and n-uplex), namely:

- ETH-SND/RCV, ETH for Gigabit Ethernet in send, receive and n-uplex;
- IBC-SND/RCV, IBC for Infiniband connected in send, receive and n-uplex;
- IBD-SND/RCV, IBD for Infiniband datagram in send, receive and n-uplex.

For the energy measurements, I use Voltech PM1000+[5] power analysers (with 0.2% accuracy) connected to the machines' AC side and capable of reading the power twice per second. For each benchmark, I select the input parameters to produce an execution time of at least 50 seconds, which allows to have at least 100 readings in each execution. Table 3.4 summarises the experimental parameters. The *data* and *time* columns denote the termination condition of each benchmark experiment. When the data size is set, the experiment terminates after transferring the indicated amount of data (i.e. the session and transport overheads), while when the timeout is set, the experiment is terminated after the indicated time. The *payload* indicates the size of the useful data in each packet, computed as a percentage of MTU minus 40 bytes (the size of IP and TCP headers), but for simplicity I denote it as "a percentage of MTU". The *connections* column indicates the number of concurrent connections through which the transfer is made. Finally, the *burst* and *throttle* represent the concrete time intervals of continuous activity and inactivity of the NICs. For the PSIZE benchmarks, I vary the maximum payload between 30% and 100% of the NICs' MTU. I also set the `TCP_NODELAY` flag to prevent packets smaller than MTU from being buffered. For PSIZE-DATA I set the data size to 75GB, while for PSIZE-TIME I set a timeout of 5 minutes. For the n-UPLEX benchmark, I transmit a fixed amount of data

[5] `http://www.farnell.com/datasheets/320316.pdf`

Benchmark	*Size* [GB]	*Time* [m]	*Payload* [% MTU]	*Connections*	*Burst* [ms]	*Throttle* [ms]
PSIZE-DATA	75	–	**30 – 100**	1 HD	–	–
PSIZE-TIME	–	5	**30 – 100**	1 HD	–	–
n-UPLEX	150	–	100	**1 – 8 FD**	–	–
PATTERN-B	11	–	100	1 HD	**1 – 10**	10
PATTERN-T	11	–	100	1 HD	10	**1 – 10**

Table 3.4: Benchmark summary with focus metric in bold.

of 150GB (sending 75GB and receiving 75GB) over n FD connections. For both PATTERN benchmarks, I set the data size to only 11GB, as the studied traffic patterns considerably increase the transfer times. In the PATTERN-B benchmark, I keep the throttle size constant to 10 ms and vary the burst size to 2, 4, 6, 8, and 10 ms. Conversely, for the PATTERN-T benchmark, I vary the throttle to 2, 4, 6, 8, and 10 ms with a constant burst size of 10 ms. I run each experiment for ten times, which ensures an average coefficient of variation of 0.053, and present the average of the results.

3.6.2 VM migration experimental setup

In this section I describe the configuration of the PMs and of the VMs I used for my evaluations. To perform these experiments, I use two hosts capable of running Xen virtualization software. These hosts should also be configured to run a VM migration between them. Another requirement for their configuration is that I should be able to measure the energy consumption of the migration, excluding the impact of load that is not generated by the experiments. I also need the machines to run a version of Xen that allows to perform both live and non-live migration. I deploy two machines and a networking switch, whose concrete hardware and software specifications are shown in Table 3.3. I use Xen version 4.2.3, that includes both xm and xl toolstacks.

The configuration for each VM is summarised in Table 3.5b. I need VMs to (1) generate load on the source/target host and (2) migrate, in order to perform the experiments. Therefore, I define two categories of VMs: `load`

and `migrating`. The load VMs are used to generate load on source or target host, while the migrating are migrated between source and target host. I employ paravirtualized VMs, because they ensure a near-native performance and are the most likely to be found in modern data centres. Each VM except the dom-0 has a dedicated storage, accessible from both PMs through NFS. I choose 4GB as size for the RAM memory because this gives a long enough migration time to clearly identify energy consumption phases. The instances `load-cpu` and `migratingx4-cpu` are used to run CPU-intensive workloads. I allocate to them four virtual CPUs to allow them to use 25% of the total CPU available on each PM. Concerning `migrating-mem`, I use this instance to run Memory-intensive workloads. Each instance of these VMs when running is pinned on different virtual CPUs, in order to remove degradation of performance due to the sharing of one CPU by multiple VMs. Finally, `dom-0` represents the PM in the Xen abstraction.

Energy measurement methodology

I employ two Voltech PM1000+[6] power measurement devices connected to the AC side of the source and target hosts, measuring their instantaneous power drawn at a frequency of 2 Hz in order to capture the power consumption of a complete VM migration, including the pre- and post-migration execution phases. For each experimental run, I start measuring the hosts' power consumption and issue a VM migration only after the measured values stabilise. Similarly, I stop the measurements after the power consumption of the hosts stabilises too. I say that the power consumption of the host stabilises when I read twenty consecutive power measurements with a difference lower than 0.3%, that is below the measurement device's accuracy. Moreover, I repeat each experiment until the difference in variance between one run and the previous runs becomes less than 10%, resulting in at least ten runs for each experiment. From the power readings and the time intervals, I compute four energy metrics: initiation, transfer and activation energy of the corresponding VM migration phases (see Sections 2.3.1 and 2.4.4), and

[6] `http://www.voltech.com/products/poweranalyzers/PM1000.aspx`

Experiment	*Configuration of source host*		*Configuration of target host*		*Configuration of migrating VM*		
	CPU	RAM	CPU	RAM	instance	CPU	RAM
`CPULOAD-SOURCE`	$[0-100]\%$	5%	`idle`	5%	`migrating-cpu`	100%	5%
`CPULOAD-TARGET`	1×`migrating-cpu`	5%	$[0-100]\%$	5%	`migrating-cpu`	100%	5%
`MEMLOAD-VM`	`idle`	5%	`idle`	5%	`migrating-mem`	100%	$[5-95]\%$
`MEMLOAD-SOURCE`	$[0-100]\%$	5%	`idle`	5%	`migrating-mem`	100%	95%
`MEMLOAD-TARGET`	1×`migrating-mem`	5%	$[0-100]\%$	5%	`migrating-mem`	100%	95%

(a) Experimental design.

ID	*Number of virtual CPUs*	*Linux kernel*	*RAM*	*Workload*	*Storage size*
`load-cpu`	4	2.6.32	512MB	`matrixmult`	1GB
`migrating-cpu`	4	2.6.32	4GB	`matrixmult`	6GB
`migrating-mem`	1	2.6.32	4GB	`pagedirtier`	6GB
`dom-0`	1	3.11.4	512MB	VMM	115GB

(b) VM configurations.

Machine	*Available virtual cpus*	*Available RAM*	*Gigabit NIC*	*Gigabit switch*	*Xen version*
`m01`	32 (16×Opteron 8356, dual threaded)	32GB	Broadcom BCM5704	Cisco Catalyst 3750	4.2.5
`m02`					
`o1`	40 (20×Xeon E5-2690, dual threaded)	128GB	Intel 82574L	HP 1810-8G	4.2.5
`o2`					

(c) Hardware configuration.

Table 3.5: Experimental setup.

the total migration energy as the sum of the three metrics.

- *Migration energy*, as defined in Equation 2.38 is the total energy consumed by the host while a migration is performed. The energy value is measured in Kilojoules and is calculated integrating the power over the time interval between the instant in which the migration begins and

the instant in which the migration ends. It is obtained by summing initiation, transfer and activation energy.

- *Initiation energy*, as defined in Equation 2.39 is the total energy consumed by the host over the initiation phase. This value is measured in Joules and is calculated over the time interval between the instant in which migration begins and the instant in which transfer phase begins.

- *Transfer energy*, as defined in Equation 2.40 is the total energy consumed by the host over the transfer phase. This value is measured in Kilojoules and is calculated over the time interval between the instant in which transfer begins and the instant in which transfer phase ends. I assume that transfer phase ends when the state of the VM has been transferred on the target host.

- *Activation energy*, as defined in Equation 2.41 is the total energy consumed by the host over the transfer phase. This value is measured in Kilojoules and is calculated over the time interval between the instant in which transfer ends and the instant in which the VM is running on the target host.

In addition, I also measure the CPU and memory consumption during each migration using the `dstat` tool and average the values of all executions.

3.7 Summary

In this chapter I described two parts that are crucial to the training and validation of my work. First, I described the code instrumentation framework, that allows me to get fine grain measurements of each PM component that is important for my model. Then, I described the applications that I have instrumented to get such measurements, both for network transfer and for VM migration. Combining these two parts I got the training and test sets that I used, for the training and the validation of my model. In the next sections, I describe the results of training and validation of these two parts of the model (network transfer and VM migration).

Chapter 4

Network transfer modelling

4.1 Introduction

In this chapter I discuss the results of the execution of the benchmarks that I described in Section 3.4.1. First, I give a short description of the network hardware and the software stack for each interface, then I show the results obtained through the execution of the benchmarks that I previously designed. Afterwards, I show the coefficients for the Equation 2.16, obtained through regression. Then, I show the error of the network model for the selected validation sets and for VM migration. The chapter is organised as follows. I review the network hardware and its software stack in Section 4.2. I analyse the benchmarks' results in Section 4.3 and I evaluate the accuracy of my model in a VM migration scenario in Section 4.3.6. Finally, I discuss my findings in Section 4.4.

4.2 Network hardware and software stack

In this section I describe the network hardware used in this work. I choose to use Ethernet and Infiniband NICs, because they are to the best of my knowledge the most used interconnection technologies used in data centres. While communications running on Ethernet use the implementation of TCP/IP provided by the OS, Infiniband software stack relies on kernel-bypass mech-

anisms and on RDMA-based capabilities. Such capabilities have a different impact on energy consumption. Therefore, comparing these two software stacks may give interesting insights about energy consumption of network transfers. In the next two subsections, I describe both interfaces in detail.

Ethernet Ethernet is the most popular local-area network technology. It defines several protocols which refer to the family covered by the *IEEE 802.3* standard using four data rates:

- 10 Mbps for 10Base-T Ethernet, in IEEE 802.3;
- 100 Mbps, also called Fast Ethernet, in IEEE 802.3u;
- 1000 Mbps, also called Gigabit Ethernet, in standard IEEE 802.3z;
- 10-Gigabit, also called 10 Gbps Ethernet, in standard IEEE 802.3ae.

I focus on Gigabit Ethernet because, along with the newer 10-Gigabit, it is the most used interconnection technology in data centres. The minimum frame size for Gigabit Ethernet (1000Base-T standard) is 520 bytes, while the Maximum Transmission Unit (MTU) is 1500 bytes.

Infiniband Infiniband is a popular switch-based point-to-point interconnection architecture that defines a layered hardware protocol (physical, link, network, transport), and a software layer to manage the initialisation and the communication between devices. Each link can support multiple transport services for reliability and multiple virtual communication channels. The links are bidirectional point-to-point communication channels that can be used in parallel to achieve higher bandwidth. Infiniband offers a bandwidth of 2.5Gbps in its single data rate version used in this work for comparison with Gigabit Ethernet. TCP/IP communications are mapped to the Infiniband transport services through IP over Infiniband (IPoIB) drivers provided by the OS. An Infiniband NIC can be configured to work in two operational modes.

Datagram is the default operational mode of IPoIB described in RFC 4391 [100]. It offers an unacknowledged and connectionless service based on the unreliable datagram service of Infiniband that best matches the needs of IP as a best effort protocol. The minimum MTU allowed is 2044 bytes, while the maximum is 4096 bytes.

Connected mode described in RFC 4755 [101] offers a connection-oriented service with a maximum MTU of 2GB. Using the connected mode can lead to significant benefits by supporting large MTUs, especially for large data transfers.

Setting Infiniband in one of these two modes results in mapping a TCP communication on a different Infiniband transport service. For this reason, I measure the energy consumption of an Infiniband network transfer in both modes.

4.3 Experimental results

In this section I present the results of my experiments.

4.3.1 BASE

NIC	*Machine energy [kJ]*	*Network energy [kJ]*	*Execution time [s]*	*Average power [W]*	*Energy ×packet[mJ]*	*Energy ×byte [nJ]*
ETH-SND	291.8	6.01	674	8.92	0.11	73
ETH-RCV	291.3	5.94	673	8.80	0.10	71.9
IBC-SND	131.6	1.58	307	5.14	0.54	20.0
IBC-RCV	133.0	2.26	308	7.36	0.78	28.8
IBD-SND	182.1	6.33	414	15.3	0.16	78.3
IBD-RCV	175.6	5.72	401	14.3	0.14	71.0

Table 4.1: BASE benchmark results (I).

I observe in Table 4.1 a considerable difference in energy consumption for running the BASE benchmark. The immediate finding is that transferring the same quantity of data over Infiniband in connected mode is more efficient

Configuration	*Average power [W]*	*Energy per packet sent/received [mJ]*	*Energy per byte sent/received [nJ]*
ETH-SND	8.92	0.11	73
ETH-RCV	8.80	0.10	71.9
IBC-SND	5.14	0.54	20.0
IBC-RCV	7.36	0.78	28.8
IBD-SND	15.3	0.16	78.3
IBD-RCV	14.3	0.14	71.0

Table 4.2: BASE benchmark results (II).

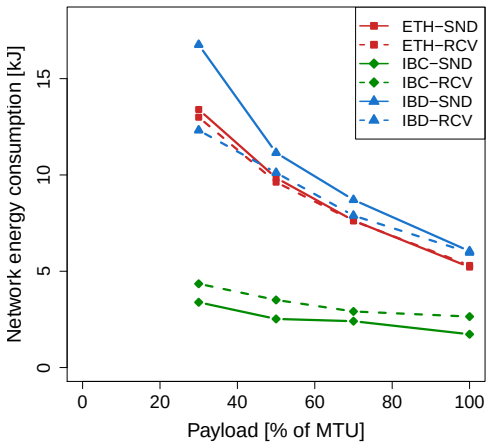
than the other alternatives in terms of energy and time. I can also observe that Infiniband's energy consumption significantly differs between sending and receiving operations: 30% less energy for sending than receiving in connected mode, and 10% less energy for receiving compared to sending. It is also noteworthy that, even in this simple benchmark, the network energy consumption is between 1.58 and 6.33 kJ, which can potentially be up to 20% of energy consumption in a node with lower idle power consumption. The other metrics provide supplementary insight into these NICs' energy efficiency. Although it might appear that the Infiniband in connected mode is more energy efficient with the lowest average power in operation, this only holds true when the two communicating parties require large amounts of on-hand data to be transferred. When the communication is message centric and the volume of effective data is low, resulting in a high number of packets being transmitted, the Gigabit Ethernet NIC is the more energy-efficient choice, closely followed by Infiniband in datagram mode. In conclusion, these preliminary findings hint that an energy efficient network communication depends on the nature of the traffic generated by the application. For data intensive traffic in applications such as data warehousing and content streaming or delivery, the more energy-efficient network is Infiniband configured in connected mode. On the other hand, for finer-grained traffic and real-time message exchanges such as low traffic databases and online games, Gigabit Ethernet is more efficient. The following experiments give further assessment of the energy consumption of the two networks with respect to traffic characteristics.

4.3.2 PSIZE

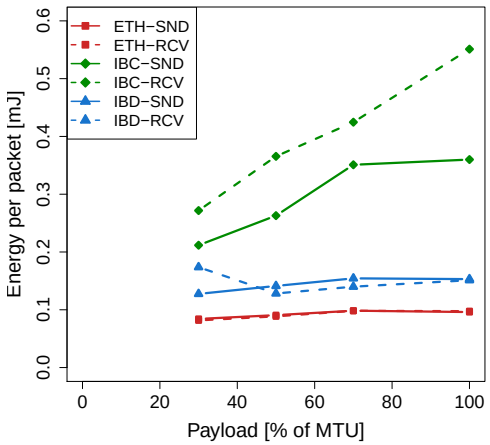
I begin with the PSIZE benchmark, focused on the influence of the payload size on networks' energy efficiency.

PSIZE-DATA. This benchmark evaluates the impact of payload on energy consumption, when sending a fixed amount of data of 75GB with a payload of 30% to 100% of MTU. The results in Figure 4.1a show that the energy consumption of the software stacks of the studied NICs is inversely proportional to payload, the most efficient operational point being reached for the maximum payload. Also noteworthy is the significantly better scalability in terms of energy when employing Infiniband NIC in connected mode: 36% energy consumption increase for a 50% decrease in payload, versus 84% for Gigabit Ethernet and 79% increase for Infiniband in datagram mode. Analysing the other metrics presented in Figure 4.1, I can identify in detail the energy-to-payload relation. Figure 4.1b suggests that, while for Infiniband in connected mode the energy consumption per transferred packet is proportional to its payload, it is relatively constant in the case of Infiniband in datagram mode and Gigabit Ethernet. Conversely, Figure 4.1c reveals a stronger inverse correlation between the payload and the energy consumption per transferred effective byte. The Infiniband in datagram mode and the Gigabit Ethernet NICs are affected in terms of energy efficiency by a payload decrease, the energy consumption per effective byte nearly tripling at a 30% of MTU payload. This behaviour is less severe for Infiniband in connected mode, the energy per byte doubling for a payload of 30% of MTU.

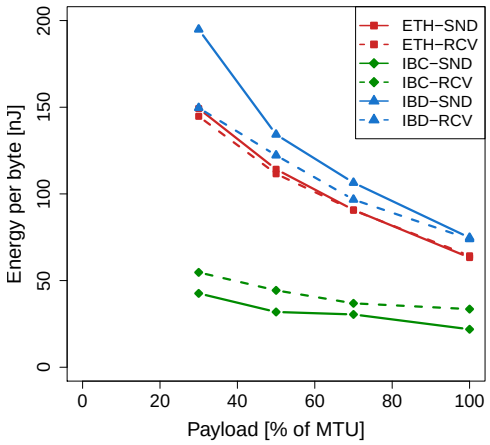
PSIZE-TIME. I present the resulting average power consumption in Figure 4.1d, each point representing the cumulated power for send and receive operations. The main finding of this experiment is that the energy consumption of both Infiniband and Gigabit Ethernet NICs is not exclusively correlated with running time. I observe that while Infiniband (regardless of its operational mode) consumes in average less power with lower payloads, Gigabit Ethernet is more power efficient at higher payloads. Further investigation revealed that Gigabit Ethernet's high power efficiency for larger



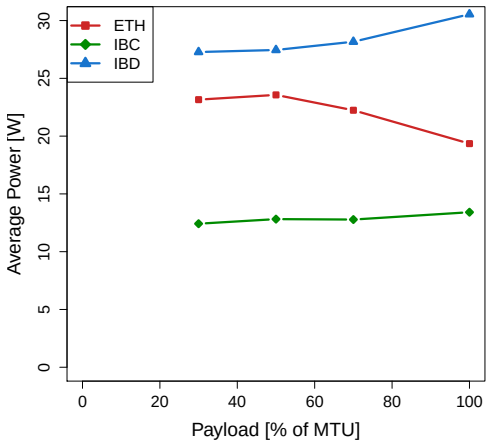
(a) PSIZE-DATA: energy consumption.



(b) PSIZE-DATA: energy per packet.



(c) PSIZE-DATA: energy per byte.



(d) PSIZE-TIME: Cumulated send and receive average power.

Figure 4.1: PSIZE benchmark results.

payloads is likely due to driver optimisations, as I noticed a 32% decrease in CPU utilisation between the transfers with payloads set at 30%, respectively 100% of MTU. The CPU utilisation was constant for all Infiniband transfers in both modes.

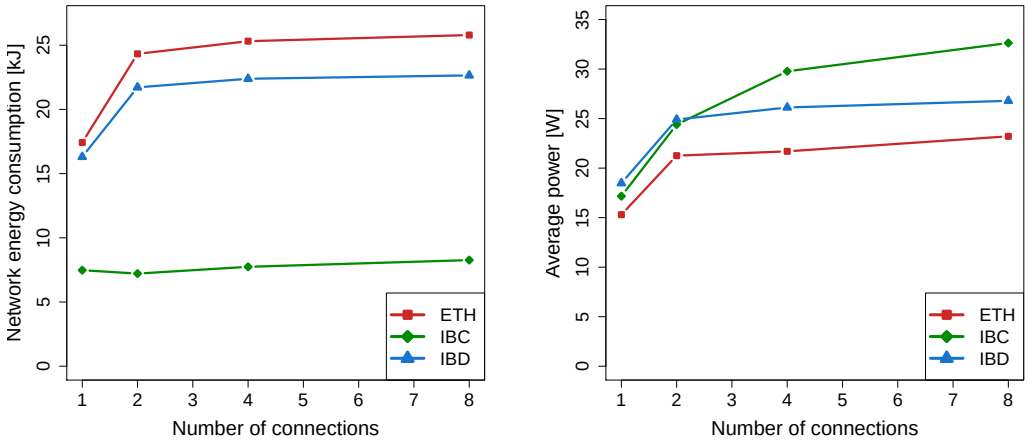
To conclude, energy consumption of the networks is inversely proportional

to the maximum payload size. Second, Gigabit Ethernet and Infiniband in datagram mode are better suited for lightweight, mixed traffic (with varying payload sizes), while Infiniband connected is by far the most energy efficient for non-fragmented traffic. Finally, network energy consumption is not exclusively time-related, thus one cannot optimise for time and expect proportional savings.

4.3.3 n-UPLEX

I observe in Figure 4.2a a considerable increase in the energy consumption of Gigabit Ethernet and Infiniband in datagram mode with more concurrent connections. The trend has a piecewise linear shape and is relatively similar for the power traces shown in Figure 4.2b. In contrast, Infiniband in connected mode shows a decreasing energy consumption with the increase in concurrent connections. Moreover, although Infiniband in connected mode consumes the least energy for transferring the fixed data amount for multiple connections, it is clearly exhibiting the highest average power consumption. This raises a question regarding the NICs' performance in terms of transfer bandwidth in this contention scenario. I present in Table 4.3 a comparison between the variation of the achieved bandwidth, consumed energy, and CPU utilisation between the two extreme cases studied: (1) the network contention case with eight concurrent FD connections and (2) the single FD connection. The results reveal a significant increase of 72% in bandwidth for the Infiniband connected, with a 19.1% average power increase. This variation of its power state with performance (in terms of bandwidth), is the reason of its energy efficiency. At the other end, Gigabit Ethernet exhibits the highest increase in energy consumption of almost 50% with only a marginal 2.5% increase in bandwidth. The considerable average power consumption increase in all cases stems from both (1) NICs requiring more power to handle the increased load and (2) increasing CPU overheads for managing multiple simultaneous connections. This observation is supported by the non-proportional energy consumption versus the CPU utilisation increase shown in Table 4.3. Finally, the increase of CPU utilisation for Infiniband in

connected mode is 130.15% higher than the other two configurations due to the increased bandwidth requiring faster data preprocessing.



(a) Network energy.

(b) Average power.

Figure 4.2: n-UPLEX benchmark results.

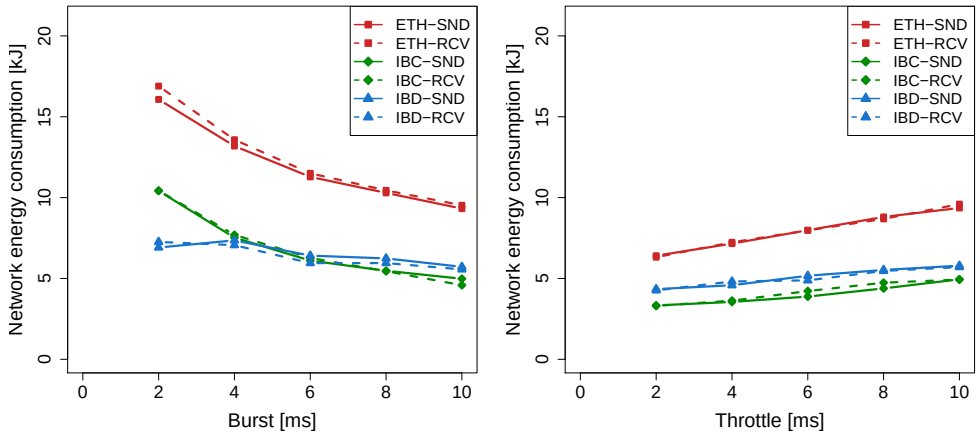
Metric	*Variation [%] (8 vs 1 connections)*		
	ETH	*IBD*	*IBC*
Bandwidth	+2.49	+4.39	+72.03
Energy	+45.80	+37.33	−31.03
Power	+49.43	+43.37	+19.11
CPU	+38.62	+38.23	+130.15

Table 4.3: Variation of relevant metrics with number of concurrent connections.

In summary, in a connection concurrency environment significant power consumption penalties occur, the Infiniband in connected mode being the best choice in terms of energy efficiency. The increased power consumption is due to a higher NICs' power state and to processing overheads.

4.3.4 PATTERN

These two experiments study the energy consumption of the NIC software stacks for different communication patterns.



(a) PATTERN-B. (b) PATTERN-T.

Figure 4.3: PATTERN benchmark results.

PATTERN-B This benchmark evaluates the impact on energy of transfers with varying burst intervals between $2 - 10$ ms with a constant throttle of 10 ms. Figure 4.3a shows that Gigabit Ethernet is the least energy efficient for all studied burst intervals. For short burst intervals (2-4 ms), Infiniband datagram is surprisingly more efficient consuming up to 44% less energy than in connected mode. For longer burst intervals, connected mode becomes better consuming 17% less energy. for the 10 ms burst pattern.

PATTERN-T This benchmark evaluates the impact on energy of transfers with a constant burst of 10 ms and a varying throttle between $2 - 10$ ms. Figure 4.3b shows a stable, monotonously increasing energy consumption with increasing throttle intervals. It is noteworthy that the energy consumption increases at different rates for the different NICs and operational

	δ_s [mJ]	δ_r [mJ]	θ_s	θ_r	ι_s	ι_r	$\mathcal{K}_s$ [kJ]	$\mathcal{K}_r$ [kJ]	ϵ	ζ	MAE $[kJ]$	$RMSE$	$NRMSE$
ETH	73.5	71.3	19.8	21.6	0.6	0.6	0.35	0.35	733.14	-685.56	0.44	0.9	0.03
IBC	137.1	181.4	13.9	14.2	0.2	0.2	0.6	0.8	12.6	-0.2	0.8	2.6	0.09
IBD	97.9	69.0	4.1	3.9	0.2	0.2	2.4	2.2	99.5	-82.1	0.8	0.9	0.05

Table 4.4: Model parameters and error.

modes: Gigabit Ethernet's consumption increases by 110 J per ms of throttle, while Infiniband by 49 J in datagram mode, and by 55 J in connected mode. Although Infiniband connected is more energy efficient for the studied configurations, a basic extrapolation shows that for traffic patterns with throttle intervals higher than 50 ms (i.e. a 1:5 burst to throttle ratio) the datagram mode becomes the more energy efficient choice.

In conclusion, Infiniband in datagram mode shows the least variation in energy consumption with different transmission patterns (good choice for mixed/undetermined transmission patterns), while Infiniband in connected mode exhibits a very good energy efficiency in a few particular cases (good choice for long transmission bursts).

4.3.5 Model evaluation

I decided to use regression analysis, that has been successfully used in previous energy prediction and modelling works [102]. I employ the NLLS regression algorithm. For extracting model parameters, I employ the data gathered from ten experimental runs. I assess the accuracy of the models using two metrics: (1) mean absolute error (MAE) and (2) root mean squared error (RMSE) that is also an absolute deviation metric, but more sensitive to large deviations. The difference between the two metrics is a measure of the variance in the individual deviations for all samples. I also present a normalised value of RMSE (NRMSE) for metric-independent comparisons. Table 4.4 shows the model parameters along with the error, calculated over all the samples. The error is always below 9.4% which demonstrates a good accuracy.

Configuration	$\delta_{\mathtt{s}}[\mu J]$	$\delta_{\mathtt{r}}[\mu J]$	$\mathcal{K}_{\mathtt{s}}[kJ]$	$\mathcal{K}_{\mathtt{r}}[kJ]$	*MAE [J]*	*NRMSE*
ETH	65.3	62.9	0.044	0.044	0.4192	0.1344
IBC	309.5	309.2	0.166	0.167	1.5015	0.1628
IBD	132.8	108.2	0.063	0.137	1.4076	0.0939

Table 4.5: Non-live migration model parameters and errors.

4.3.6 Using network transfer model for VM migration

After assessing the accuracy of the model in predicting network energy consumption of my own benchmarks, I evaluate its accuracy in predicting network energy consumption of VM migration on different NICs. I compare the measured energy value with the value that I compute employing Equation 2.16 with the coefficients of Table 4.5. I use the same hardware configuration as in Table 3.4, and a `dom0` GNU/Linux kernel version 3.0.4 for running Xen on one CPU with 512MB of RAM. I migrate a paravirtualized VM running a 2.6.32 Linux kernel on one CPU, and set its memory size to 4GB to ensure a long-enough migration time for an accurate energy measurement. I issue the migration by using Xen's `xm` command line interface. I measure the network energy consumption by instrumenting the machines during the migration time and subtracting their static energy consumption. I employ again Equation 2.16 to calculate VM migration energy consumption. To determine t_{end} (Equation 2.25) I set the $\mathtt{DATA}_{\mathtt{send}}$ and $\mathtt{DATA}_{\mathtt{receive}}$ parameters for the SND and RCV configurations to the memory size in bytes. I set $b_{\mathtt{send}}$ and $b_{\mathtt{receive}}$ to the migration time, $t_{\mathtt{send}}$ and $t_{\mathtt{receive}}$ to 0, and finally $c_{\mathtt{send}}$ and $c_{\mathtt{receive}}$ to 1. I extracted new α and $\mathcal{K}$ parameters because of the different kernel version. In Table 4.5 I show the estimation error for four runs (average coefficient of variation of 0.012). I use the range of values of each execution as range for NRMSE. I observe that my model has a maximum MAE of 1.5J, which corresponds to a 16.2% NRMSE. In most of the cases, anyway, the NRMSE is below 9.4%, with a MAE lower than 0.9J, which compared to the total energy consumption for migration (between 248J and 349J) is a quantity which does not significantly affect the accuracy of the prediction.

Application characteristic	*Preferred NIC*
Big data, continuous traffic/ *non-live migration*	Infiniband connected
Continuous message passing *live migration (low dirtying rate)*	Gigabit Ethernet
Multiple parallel connections	Infiniband connected
Low communication/computation *live migration (low dirtying rate + high cpu usage)*	Infiniband datagram
High communication/computation *live migration (high dirtying rate)*	Infiniband connected

Table 4.6: Guidelines for NIC selection depending on communication characteristics.

4.4 Discussion

These results show that there is no "best" NIC in terms of energy efficiency. Furthermore, even setting the same NIC to different operational modes produces distinct results. Comparing for example the results from the PSIZE (Section 4.3.2) and PATTERN (Section 4.3.4) benchmarks, I found out that Infiniband connected outperforms Infiniband datagram for continuous data transfers at maximum payload (72% less energy consumption), while for different communication patterns Infiniband in datagram mode is 44% more efficient than in connected mode. Therefore, choosing the optimal NIC for energy efficient communication depends on the application requirements and its communication characteristics. For exchanging large data quantities, Infiniband connected saves over 50% of energy compared to Gigabit Ethernet or Infiniband datagram. However, if the application needs to frequently send or receive small packets, Infiniband operating in datagram mode can be a better choice. When the number of exchanged messages is more relevant to the application than the quantity of data transferred, Gigabit Ethernet presents the lowest energy consumption per transferred packet.

One could think on dynamically exploiting NIC capabilities by selecting at runtime the most energy-efficient interface for the given application's communication characteristics. Multiple applications contending for NICs

can also contribute to the complexity of this task. As a first step towards this challenging goal, I use this work to define the general guidelines for making a correct decision from an energy efficiency perspective based applications communication characteristics, as summarised in Table 4.6. Concerning VM migration, as I previously observed in Figure 2.2 and in the description of the VM migration process in Section 2.3.1, the only phase in which network transfer is involved is the transfer phase. In this phase, the state of the VM is transferred over the network. The way such transfer is performed is dependent on the type of VM migration performed and on the type of workload VM is executing. For example, in non-live migration the state of the VM is sent over the network in a single transfer, whose length depends on (1) the amount of VM RAM and (2) the bandwidth between $\mathcal{S}$ and $\mathcal{T}$. However, things changes if a live migration is performed. In fact, traffic pattern may change due to the continuous update of the VM state that is necessary to perform the live migration. Therefore, there is no "better" NIC to perform VM migration, and the only difference between energy consumption is due of the bandwidth available. Therefore, my evaluations are performed on Gigabit Ethernet, as at the time of writing it is one of the most used interconnections technologies.

4.5 Summary

In this chapter a comparative analysis of the energy efficiency of today's mostly used NIC families in data centres, Gigabit Ethernet and Infiniband. Analysis is performed by developing benchmarks aiming at stressing different network transfer parameters. After performing the analysis, I use 20% of the data collected in this phase as training set for the model I designed in Section 2.4.2. Then, I evaluate the accuracy of the model for the rest of data I collected and then I use the same model to predict energy consumption of VM migration, showing that my model is capable to achieve an error between 3 and 9% for network transfers and between 9.3 and 16.2% for VM migration. The slight increase in the error is because, when using this model to predict energy consumption of VM migration, I am just modelling the

network transfer part, ignoring the energy consumption due of the different VM migration phases identified in Section 2.3.1. Thus, while this model showed its accuracy for network transfers, it is still not enough to model VM migration as not only energy consumption of transferring VM state over the network has to be considered, but also the other parameters I identified in Section 2.4.4. In the next section, I evaluate my model for VM migration built over the network transfer model I validated in this chapter.

Chapter 5

VM migration modelling

5.1 Introduction

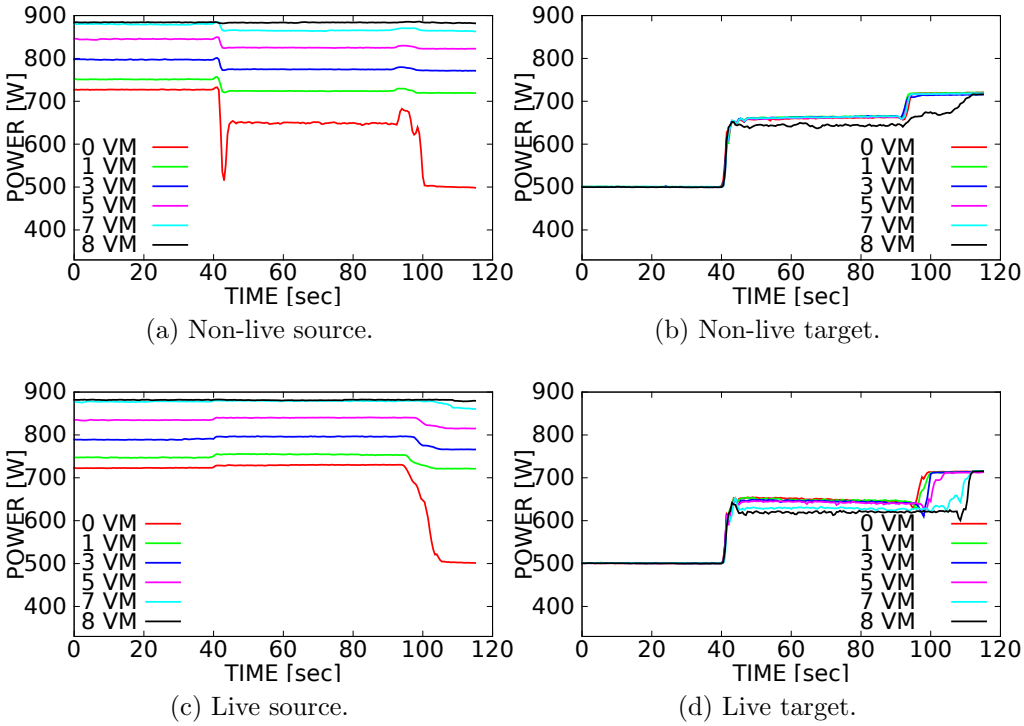
In this chapter, I discuss the results of the benchmarks' execution for VM migration. The benchmarks that I execute are the ones I described in Section 3.5. First, I describe the power draw during the execution of each benchmark, showing how the power draw varies through the execution of each phase identified in Section 2.3.1. Then, I show the coefficients obtained after the regression analysis, and use them to compute energy consumption of VM migration, employing Equation 2.38. For power draw, I use instead Equations from 2.44 to 2.46. Afterwards, I compare the results obtained with my model with other state-of-the-art models for VM migration. The chapter is organised as follows. I present the results of my experimental validation in Section 5.2 based on the benchmarking methodology described in Section 3.5. Finally, I perform a comparison with other state-of-the-art models in Section 5.3.

5.2 Experimental results

In this section, I show the results of the experiments described in Section 3.5. For each experiment I report the instantaneous power consumption traced every 500 milliseconds (according to the resolution of the power measure-

ment devices) which allows to easily identify the migration phases. I extract the energy consumption for each phase by integrating the power over its length. I average each result over ten experimental runs to ensure statistical significance.

5.2.1 CPULOAD-SOURCE



(a) Non-live source.

(b) Non-live target.

(c) Live source.

(d) Live target.

Figure 5.1: CPULOAD-SOURCE results.

The results for the CPULOAD-SOURCE experiment displayed in Figures 5.1a and 5.1b show that the instantaneous power consumption of a non-live migration follows the same trend for each CPU workload except the case with eight VMs, when I have multiplexing on the machine's CPUs. In this case, I clearly see that on the source host (Figure 5.1a) the power consumption trend follows a constant function, since it is proportional to the CPU usage that never exceeds its hardware-imposed limit beyond which the resources are shared between the VMs. In this case, the migrating VM is

suspended when the migration starts and the load on the host drops when there is no multiplexing without affecting the power consumption.

Concerning the target (Figure 5.1b), I notice a slightly lower power consumption from the beginning of the transfer phase when the source host has full CPU utilisation because of the reduced bandwidth to the target host (due to the 100% CPU load on the source host). A reduced bandwidth implies a lower power consumption and a longer transfer phase.

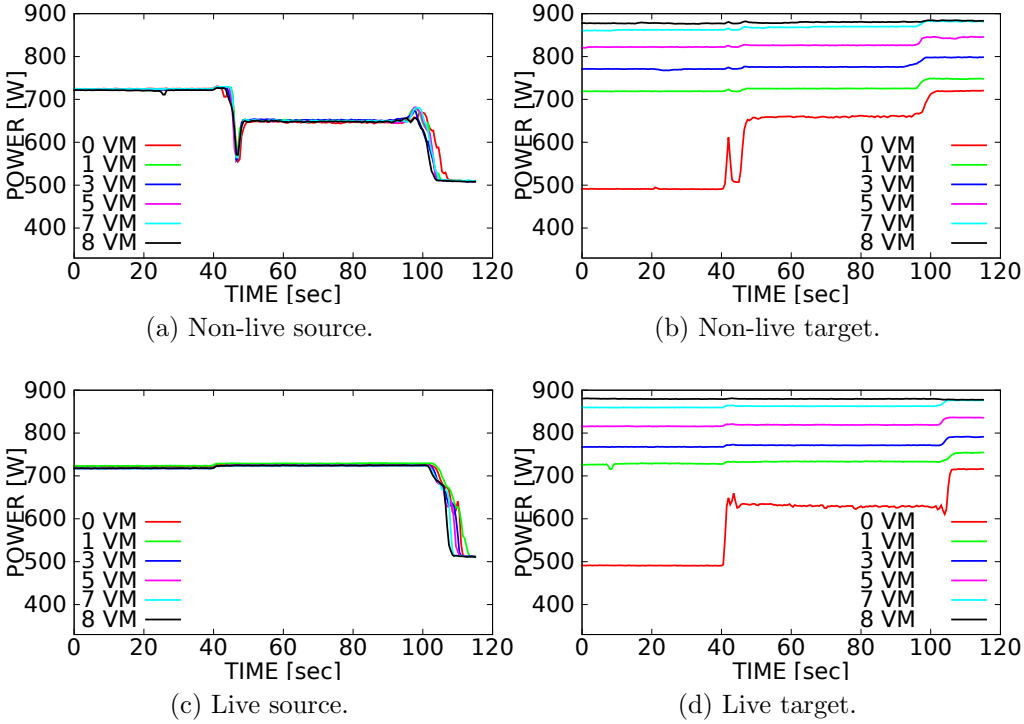
For live migration (Figures 5.1c and 5.1d), I observe an increased power consumption over the transfer phase due to the running VM because of: (1) the additional power consumption for network transfers and (2) the increased CPU usage of the virtualization software to handle the live migration. Concerning the source host, I notice a constant power consumption in case of CPU multiplexing, for the same reason as in Figure 5.1a.

Considering the power consumption on the target host (Figure 5.1d), I observe no significant differences compared to the non-live migration, except for a reduced consumption for the full CPU load with and without multiplexing. This is because the migrating VM is not suspended over the transfer phase and thus, it still uses CPU resources on the source host. Therefore, the source host is not able to exploit the full bandwidth available between the two hosts, leading to a scenario similar to the one observed in Figure 5.1b. I also notice a strong difference in power consumption before and after the migration in the 25% load scenario because the power drawn of the source host returns back to idle after the migration.

I conclude that CPU-intensive workloads have an impact on VM migration when running on the source, as bandwidth decreases when the CPU is fully loaded causing a longer transfer phase and a consequently, a higher energy consumption.

5.2.2 CPULOAD-TARGET

For the CPULOAD-TARGET experiment, I observe first in Figure 5.2a that the impact on the power consumption of source host is minimal when changing the load on the target. Concerning the target measurements in Fig-



(a) Non-live source.

(b) Non-live target.

(c) Live source.

(d) Live target.

Figure 5.2: CPULOAD-TARGET results.

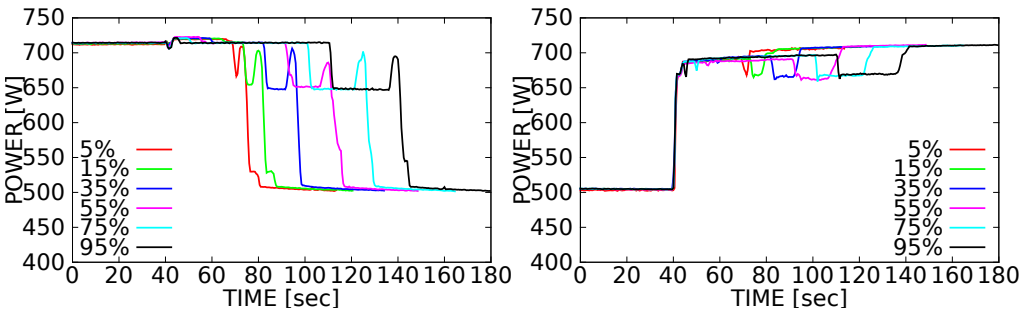
ure 5.2b, I notice (1) a small increase in power drawn due to the network transfer of the VM state and (2) a big increase in the power consumption when the migration is finished and the VM is up and running on the target. The impact of external load in this case is visible only when the target host is fully loaded, where the power resembles a constant trend since the host reached its CPU limit (see Equation 2.31).

For live migration (Figure 5.2c), I notice for the source host a small increase in power consumption over the transfer phase due to: (1) the network transfer of the VM state and (2) the CPU increase for handling the migration. I do not notice any impact of the target load on this host except for the slight difference in case of multiplexing due to the additional load on the target host that prevents the VMM to use the full bandwidth. For the target host in Figure 5.2d, I see similar trends to the non-live migration except that: (1) the power drawn is slightly lower in the transfer phase and (2) the live migration takes at least 60 seconds longer. However, since this tendency is present also

in the idle target case, it seems mostly related to hardware configuration than the host load.

5.2.3 MEMLOAD-VM

For the MEMLOAD-VM experiment, I observe in Figures 5.3a and 5.3b that the power consumption considerably changes with the dirtying ratio, with the difference that for the target host it does not go back to the idle level but slightly increases (since the VM is running on the target afterwards). On both hosts, the drop in power consumption during the transfer phase grows with the dirtying ratio because the VM experiences a longer suspension time to complete the migration by sending the more dirty memory pages from source to target.



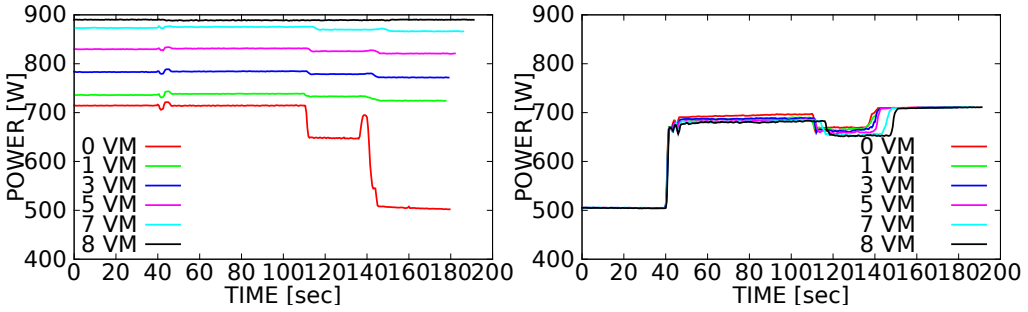
(a) Source. (b) Target.

Figure 5.3: MEMLOAD-VM results.

5.2.4 MEMLOAD-SOURCE

For the MEMLOAD-SOURCE experiment, I observe in Figure 5.4a that the transfer phase increases with the CPU load on the source host and the memory-intensive workload running on the VM. This slight increase is proportional to the decrease in bandwidth utilisation due to the increased CPU usage of the source. This tendency is better seen for high amount of loads for the target host (Figure 5.4b), when I notice a considerable increase in the transfer phase due to the reduced bandwidth. I also observe that the CPU

load on the source host has an impact on the energy consumption of migration even in case of memory-intensive workloads, for which reason I included it in Equation 2.45. Finally, I also notice on both hosts a considerable drop in power consumption towards the end of the transfer phase because of the VM suspension on the source due to the high dirtying ratio that transforms the live migration in a non-live one (i.e. the VMs are not accessible from the network during this time). The similarity with non-live migration is clear by looking at Figures 5.1a and 5.1b.



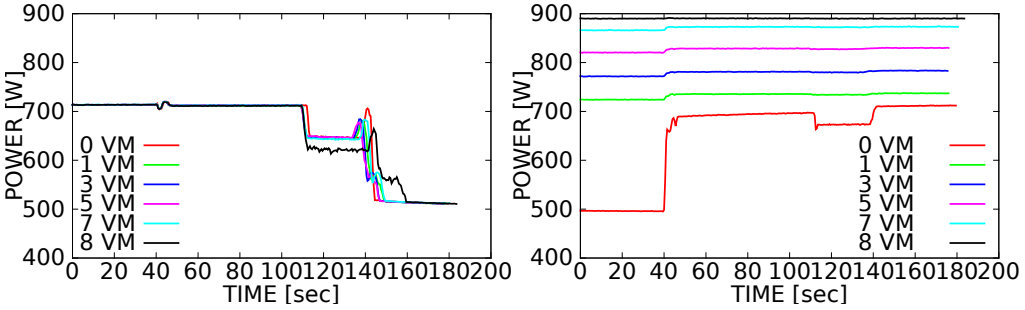
(a) MEMLOAD-SOURCE, measurements on source host

(b) MEMLOAD-SOURCE, measurements on target host

Figure 5.4: MEMLOAD-SOURCE results

5.2.5 MEMLOAD-TARGET

For the MEMLOAD-TARGET experiment, I see in Figure 5.5a that the transfer phase has a similar length on the source host, except for the slight difference in case of multiplexing due to bandwidth limitations on the target. The trends of the activation phase assume a different shape according to the amount of load. On the target host (Figure 5.5b), I observe a constant trend in power consumption except the idle case, when live migration becomes a non-live one as I can see by comparing the highlighted areas in Figures 5.2a and 5.2b.



(a) MEMLOAD-TARGET, measurements on source host

(b) MEMLOAD-TARGET, measurements on target host target

Figure 5.5: MEMLOAD-TARGET results

5.2.6 Regression analysis

In this section I use linear regression to compute the model coefficients I identified in Section 2.4.4. For each phase identified in the theoretical model (see Section 2.3.1) I use regression analysis based on the Non Linear Least Square algorithm. Afterwards, I use the coefficients obtained for the power functions defined in the Equations from 2.44 to 2.46. I select a training subset of the power readings from each phase to extract the model coefficients and use them afterwards as a model to predict the energy consumption. The training set used for this purpose is the 20% of the readings obtained by running the experiments on the machines `m01` – `m02`. The coefficients for non-live migration are summarised in Table 5.1, while the coefficients for live migration are summarised in Table 5.2. To validate my model, I also used the same coefficients to predict the energy consumption of non-live and live migration on a different set of machines (`o1` – `o2`). When checking the results of the prediction on this new set, I observed that it was overestimating the measured values by a constant factor because the bias obtained from the training phase includes the idle power consumption of the PM. Therefore, I changed the bias by subtracting the difference in idle power between the two sets of machine. I use then C^m as bias for the prediction on (`m01` – `m02`) and C^o for the prediction on (`o1` – `o2`). The error for this model in both datasets is shown in Table 5.3. The discussion of my model's behaviour is presented

in contrast to other state-of-the-art models in the next section. In the next section, I show the error for energy consumption obtained with the obtained coefficients.

Host	*Initiation*				*Transfer*				*Activation*			
	$\alpha_{(\mathtt{i})}$	$\beta_{(\mathtt{i})}$	$\mathcal{C}^m_{(\mathtt{i})}$	$\mathcal{C}^o_{(\mathtt{i})}$	$\alpha_{(\mathtt{t})}$	$\beta_{(\mathtt{t})}$	$\mathcal{C}^m_{(\mathtt{t})}$	$\mathcal{C}^o_{(\mathtt{t})}$	$\alpha_{(\mathtt{a})}$	$\beta_{(\mathtt{a})}$	$\mathcal{C}^m_{(\mathtt{a})}$	$\mathcal{C}^o_{(\mathtt{a})}$
$\mathcal{S}$	1.71	1.41	708.3	165	2.4	1.08[1]	421.74	200	2.37	0	662.5	150
$\mathcal{T}$	3.18	0	596.06	162	2.56	5.49[2]	520.214	210	1.88	17.01	499.56	100

[1] $\times 10^{-6}$
[2] $\times 10^{-7}$

Table 5.1: Coefficients for non-live migration.

Host	*Initiation*				*Transfer*						*Activation*			
	$\alpha_{(\mathtt{i})}$	$\beta_{(\mathtt{i})}$	$\mathcal{C}^m_{(\mathtt{i})}$	$\mathcal{C}^o_{(\mathtt{i})}$	$\alpha_{(\mathtt{t})}$	$\beta_{(\mathtt{t})}$	$\gamma_{(\mathtt{t})}$	$\delta_{(\mathtt{t})}$	$\mathcal{C}^m_{(\mathtt{t})}$	$\mathcal{C}^o_{(\mathtt{t})}$	$\alpha_{(\mathtt{a})}$	$\beta_{(\mathtt{a})}$	$\mathcal{C}^m_{(\mathtt{a})}$	$\mathcal{C}^o_{(\mathtt{a})}$
$\mathcal{S}$	1.71	1.41	708.3	165	2.4	1.52[1]	1.41	0.4	421.74	200	2.37	0	662.5	150
$\mathcal{T}$	3.18	0	596.06	162	2.56	7.32[2]	0	0.4	520.214	200	1.88	17.01	499.56	100

[1] $\times 10^{-6}$
[2] $\times 10^{-7}$

Table 5.2: Coefficients for live migration.

Model	*Host*	*NRMSE* (non-live) (m01 – m02)	*NRMSE* (live) (m01 – m02)	*NRMSE* (non-live) (o1 – o2)	*NRMSE* (live) (o1 – o2)
WAVM[3]	Source	11.8%	11.8%	12.5%	12.7%
	Target	12%	5%	16.3%	17.2%

Table 5.3: Normalised root mean square error (NRMSE) of my model on the two datasets.

5.3 Comparison

In this section, I compare the accuracy of the model with three other models available in the literature that take into account different parameters to

model energy consumption of VM migration: HUANG [16], LIU [17] and STRUNK [18]. Next, I shortly describe each one of these models.

HUANG The model of Huang et al. [16] is based on the assumption that the instantaneous power consumption P of each host is linear with the CPU utilisation of the VM v at the instant t, $load_{cpu}(v,t)$ [98]:

$$P(h,v,t) = \alpha_{huang}(h) \cdot load_{cpu}(h,t) + \mathcal{C}(h), \tag{5.1}$$

where the istantaneous power $P(t)$ is linear by a factor of α and $\mathcal{C}$ is a hardware-related constant. I obtain the energy consumption by integrating P over the migration time $[m_s, m_e]$:

$$E_{migr}(h,v) = \int_{m_s}^{m_e} P(h,v,t)\ dt. \tag{5.2}$$

This model perfectly suits scenarios when CPU utilisation has an impact on VM migration, but does not suit scenarios that involve other parameters (e.g. memory dirtying ratio, CPU load on migrating VM).

LIU The model of Liu et al. [17] is based on the assumption that energy consumption for migrating VM v $E_{migr}(v)$ depends only on the amount of data $\texttt{DATA}$ exchanged by the two hosts during the VM migration:

$$E_{migr}(h,v) = \alpha_{liu} \cdot \texttt{DATA}_{liu}(v) + \mathcal{C}(h), \tag{5.3}$$

In their work, the authors compute the amount of data exchanged during migration as a function of VM memory size, memory transmission rate and memory dirtying ratio. Moreover, since they assume transfer is performed in several rounds, they compute the amount of data as the sum of the data sent in each round:

$$\texttt{DATA}_{liu} = \sum_{i=0}^{\mathcal{I}} \frac{\texttt{RAM}_{max}(v)}{\texttt{BW}_{net}(\mathcal{S},\mathcal{T})} \cdot \texttt{DR}(v,i), \tag{5.4}$$

where $\mathcal{I}$ is the number of rounds, $\texttt{BW}(\mathcal{S}, \mathcal{T})$ is the bandwidth between $\mathcal{S}$ and $\mathcal{T}$ and $\texttt{DR}(v, i)$ the dirtying ratio over the round i. I use instead the amount of data transferred measured with the network instrumentation as the `DATA` value. In this model, α_{liu} models the linear relationship between the transferred data and energy consumption and $\mathcal{C}(h)$ is a hardware-related constant. For this reason, the model is perfectly suitable for predicting the energy consumption of VMs workloads with high dirtying ratio. This model, however, does not consider the CPU load which generates modelling errors in case this has a high impact on the energy consumption. Moreover, it assumes that homogeneous hosts have the same consumption during migration. However, as stated also by [103], such an assumption could lead to inaccurate results.

STRUNK The model of Strunk [18] considers the VM memory size $\texttt{RAM}(v)$ and the network bandwidth between source and target $\texttt{BW}_{net}(\texttt{S})$ as parameters in a linear model:

$$E_{\texttt{migr}}(h, v) = \alpha_{strunk} \cdot \texttt{RAM}_{max}(v) + \beta \cdot \texttt{BW}_{net}(\mathcal{S}) + \mathcal{C}_{strunk}(h), \tag{5.5}$$

where α_{strunk} and β_{strunk} model, the linear relationship between VM size and network bandwidth and $\mathcal{C}(h)$ is a hardware-related constant. This model perfectly suits scenarios in which both hosts and the migrating VM are idle and does not take their load into account. Even though such conditions are very likely to happen in data centres [104], many works show the benefits of consolidating VMs executing tasks to/from hosts that are not idle [105]. Therefore, having a model able to predict the energy consumption of VM migration in different conditions can be helpful to decide whether this is beneficial energy-wise.

I train these models using the same training set used to train my model and the coefficients obtained for each model are summarised in Table 5.4. Afterwards, I compute three error metrics on the test set: Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Normalized Root Mean Square Error (NRMSE). Each metric is summarised in Table 5.5. In the next subsections, I compare the results of my model, named Workload-Aware

Model	*Host*	α	β	$\mathcal{C}$
HUANG	Source	2.27	–	671.92
	Target	2.56	–	645.776
LIU	Source	2.43	–	494.2
	Target	2.19	–	508.2
STRUNK	Source	3.35	−3.47	201.1
	Target	5.04	−0.5	201.1

Table 5.4: Training phase coefficients for LIU, HUANG and STRUNK models, calculated for the PM sets m01-m02 and o1-o2.

Model	*Host*	*MAE* (non-live) [kJ]	*RMSE* (non-live)	*NRMSE* (non-live)	*MAE* (live) [kJ]	*RMSE* (live)	*NRMSE* (live)
WAVM^3	Source	1.8	2558	11.8%	6.3	8432	11.8%
	Target	1.7	1789	12%	3.6	4056	5%
HUANG	Source	1.8	2587	12%	5.5	9234	15.7%
	Target	1.8	2067	12.8%	7.1	9102	12.9%
LIU	Source	4.8	5812	26.9%	9.8	12117	36.3%
	Target	3.4	4121	25.3%	7	9622	29.4%
STRUNK	Source	0.026	3824	17.7%	0.028	4547	35.4%
	Target	0.058	5187	30%	0.019	4382	36.2%

Table 5.5: Comparison of WAVM^3 with other models on dataset m01-m02.

Virtual Machine Migration Model (WAVM^3), with the other three.

5.3.1 Non-live migration

By looking at Table 5.5, I observe that, among the analysed models, the one of Huang et al. provides the most accurate estimation for non-live migration. This is because non-live migration is mostly influenced by CPU usage which is the only parameter that this model takes into consideration. Since my model also takes CPU into account, I do not expect high variations in most of the scenarios. However, it can happen that one host is not able to use the full bandwidth if there is some multiplexing on the CPU. In such situations, network utilisation drops because CPU is not able to exploit all the network resources available and, therefore, network bandwidth cannot be ignored. Since my model also takes into account network bandwidth, it manages to

have better estimations (−0.2% NRMSE for source host, −0.8% NRMSE for target host) when there is less network bandwidth available. Moreover, even though the MAE for the two models is very similar, I observe that the difference between RMSE and MAE is slightly higher for the model of Huang et al., showing that my model's estimation error has a lower variance too.

5.3.2 Live migration

The errors for the live migration are summarised in Table 5.5. Also in this case, the model of Huang et al. performs considerably better because it considers the CPU of source and target hosts, ignored by the other two, that has a considerable impact on energy consumption during VM migration. However, I notice an 18% increase in NRMSE versus the non-live migration error for the source host and a 16.2% increase in NRMSE for target host. This is because live migration should taken into account the CPU utilisation and the dirtying ratio of the migrating VM that is still running during the migration. This model performs better because these parameters are instead considered, increasing the accuracy of prediction of Huang et al. by 3.9% (11.8% vs 15.7% NRMSE) for the source host and by 7.9% (5% vs 12.9%) for the target host.

5.4 Summary

In this chapter I performed a validation of the VM migration model described in equations from 2.4.4. To perform my validation, I set up a small data centre with two sets of different PMs, with a virtualization environment based on Xen hypervisor. Then, I developed purposely designed benchmarks mimicking the workload characteristics highlighted in Section 2.3.1. Based on the measurements I collected in this phase, I used linear regression to obtain the coefficients of Equations 2.44, 2.45 and 2.46. Then I validated the model on two different test sets, obtained by measuring power consumption on two different sets of PMs. Finally, I performed a comparison with other existing models for VM migration. First, I used linear regression to get the

coefficients needed by such models, using the same training set I used for my model. Then I compared their results with the measurements in the test set, showing that my model improves their accuracy. However, in order to be useful, such model should be used for simulations of Cloud data centres. In the next chapter, I describe the implementation of this model inside a simulation environment.

Chapter 6

Integration into simulators

6.1 Motivation

In this chapter, I show how I port the previously designed models from Chapter 2 inside a Cloud simulator. Doing so allows my model to be used in more extensive cases. I implemented it in a Cloud infrastructure simulator called DISSECT-CF [106] which is integrated as a backend of the user-side GroudSim simulator [82]. I show that implementing in this simulator the model I designed in Chapter 2 increases the accuracy of the simulation of VM migration and similar major activities involved in the workload consolidation process, confirming the results of the validation process that I describe in Chapters 4 and 5. My ultimate aim is to provide the distributed systems research community with a model that is: (1) easy to implement, and (2) able to capture the behaviour of different types of data centres components.

I validated my implementation by comparing it with real-life measurements from various benchmarks executed on VMs migrated across two different sets of hosts in a private Cloud. This way, I managed to: (1) improve the energy models of GroudSim/DISSECT-CF with the help of piecewise linear regression, (2) validate my new model's implementation under different kinds of operational scenarios with and without VM migration, and (3) achieve a 45.2% improvement in normalised error (NRMSE) compared to the state-of-the-art CloudSim simulator.

The chapter is organised as follows. First, I evaluate the simulation's accuracy in Section 6.4. I describe in Section 6.3 the implementation of the model in the GroudSim and DISSECT-CF simulators, and evaluate its performance and accuracy in Section 6.4.

6.2 Model evaluation

In the next section, first I describe the training and validation for the energy model I defined in Section 2.4. I show the coefficients obtained through linear regression, as well as the validation on the real measurements, obtained using the methodology described in Chapter 3.

6.2.1 Regression modelling

Here I show the regression modelling and the validation of the system model that I use in my simulator, described in Section 2.4. While $P_{\max}(h)$ and $P_{\text{idle}}(h)$ are obtained by measuring the idle and maximum power consumptions with thr power measurement devices, the coefficients $\alpha(h)$ and $\beta(h)$ are extracted by using the CubiST [1] tool. I chose this tool because it generates rule-based regression models with respect to the model designed for CPU power consumption from Equation 2.20. I imposed to the tool to generate no more than two rules for simplicity reasons. As training set, I chose a subset of 20% of the measurements and used the rest for validation and evaluation. To improve the model's accuracy, I also employed the 10-fold cross-validation provided by the CubiST tool. I employ two error metrics: the Mean Absolute Error (MAE) and the Normalised Root Mean Square Error (NRMSE). The validation results shown in Table 6.1 using a test set of 80% of the measurements show a NRMSE below 7% for both data sets, with an average MAE of 18.28W on the {`m01`, `m02`} machines, and of 11.5W on the {`o1`, `o2`} machines. I also compared the piecewise linear prediction with the linear model in [32] showing a reduction in NRMSE of 9.4% (15.6% versus 6.2%) on {`m01`, `m02`} machines and of 7.1% (13.9% versus 6.8%) on

[1] `https://www.rulequest.com/cubist-info.html`

Model	*Machine set*	$\mathcal{L}$	α	β	γ	δ	P_{idle} [W]	$P_{\max}$ [W]	*MAE* [W]	*NRMSE* [%]
WAVM^3+DCM	m01, m02	0.12	5.29	0.68	0.05	0.1	501	840	18.28	6.2
	o1, o2	0.12	4.33	0.47	0.05	0.1	164	382	11.5	6.8
Linear	m01, m02		284.974	618.67					47.8	15.6
	o1, o2		165.08	212.563					23.7	13.9
Cubic	m01, m02		209.317	695.684					69.2	23.6
	o1, o2		140.753	235.776					37.3	20.4

Table 6.1: Model coefficients and errors.

{o1, o2} machines. The improvement is even higher compared with the cubic model in [80], with a reduction in NRMSE of 17.4% (23.6% versus 6.2%) on the {m01, m02} machines and of 13.6% (20.4% versus 6.8%) on {o1, o2}.

6.3 Simulation framework

In this section, I describe the simulation framework in which I implement the models consisting of two main parts: GroudSim, which provides the user side of the IaaS Cloud, and DISSECT-CF, which provides the internal infrastructure side.

6.3.1 GroudSim

GroudSim is a Java-based simulation toolkit for scientific applications running on Grid and Cloud infrastructures, depicted in Figure 6.1. GroudSim uses a discrete-event simulation toolkit consisting of a future event list and a time advance algorithm that offers improved performance and scalability compared to other process-based approaches [33]. The simulation framework supports modelling of Grid and Cloud computational and network resources, job submissions, file transfers, as well as integration of failure, background load, and cost models. An advanced textual and visual tracing mechanism and a library-independent distribution factory give extension possibilities to the simulator. New tracing mechanisms can be easily added by implementing new handlers or filters in the event system, and additional distribution func-

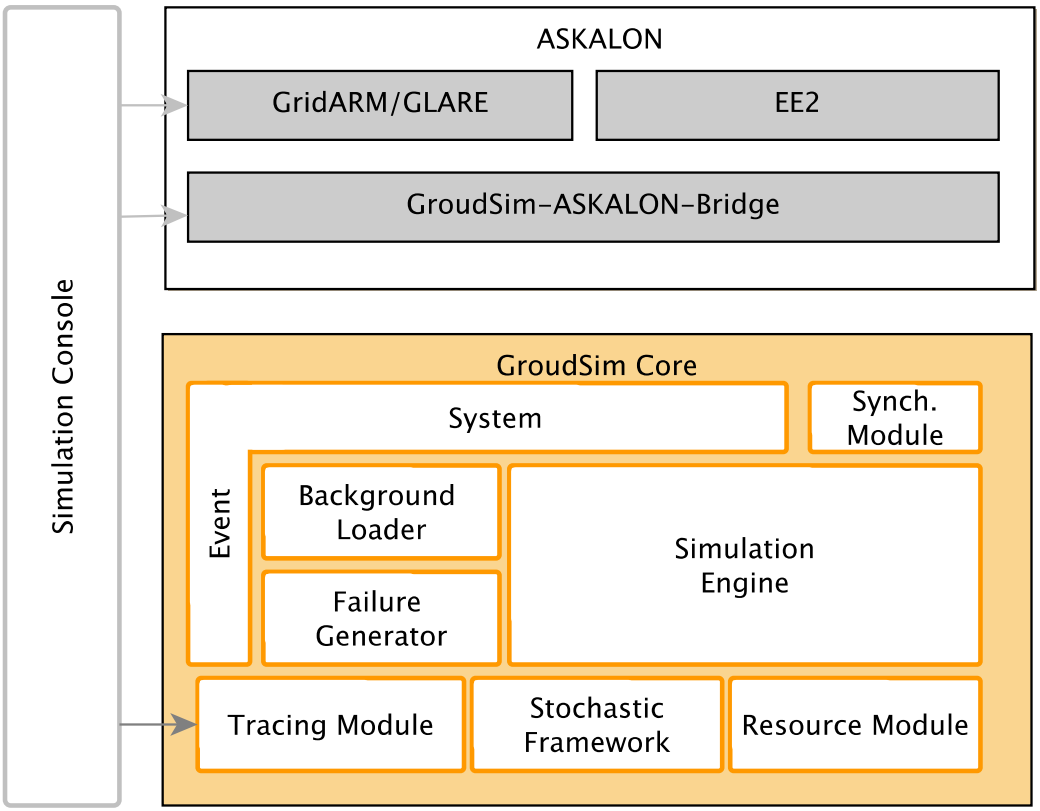


Figure 6.1: GroudSim architecture.

tions can be included by adding a new library and writing an appropriate adapter. GroudSim focuses on the user-side of IaaS Cloud computing and is currently used as an additional backend in the ASKALON system enabling users to perform seamless development, debugging, simulation and execution of Grid/Cloud applications using the same interface [83].

6.3.2 DISSECT-CF

As shown in Figure 6.1, GroudSim lacks knowledge of the internal IaaS infrastructure. Since this knowledge is essential for the simulation of energy consumption in data centres, I connected it to DISSECT-CF that is a compact and highly customisable open source Cloud simulator with special focus on the IaaS systems. Figure 6.2 summarises the DISSECT-CF architecture with its five major subsystems:

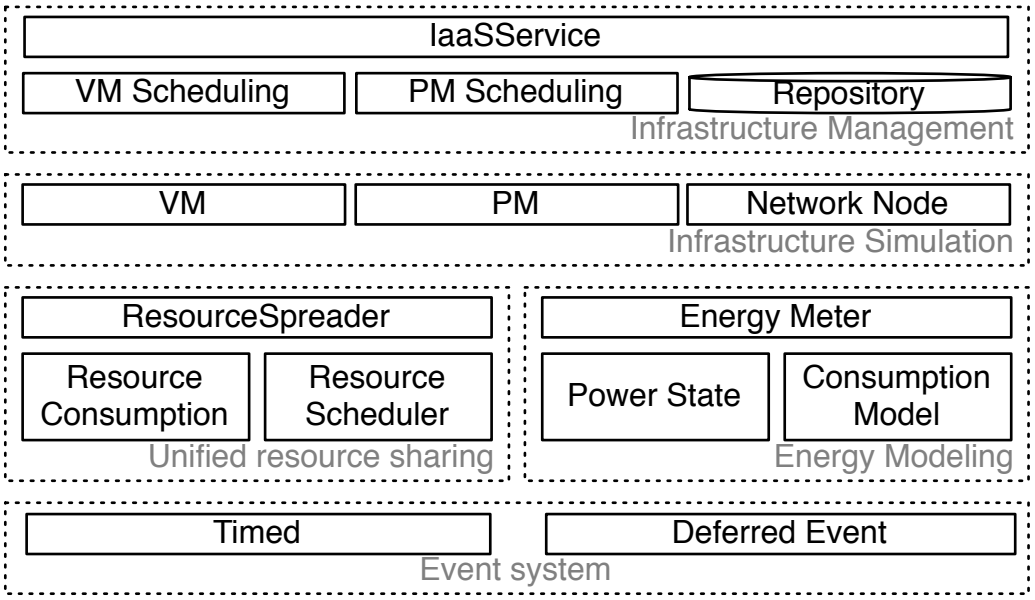


Figure 6.2: DISSECT-CF architecture.

- *Event system* for providing a unified time reference;
- *Unified resource sharing* for modelling the sharing of the data centre resources.
- *Energy modelling* for simulating energy usage patterns of individual components (CPU, network, storage);
- *Infrastructure simulation* for modelling IaaS components, such as PMs, VMs and network entities;
- *Infrastructure management* encapsulating scheduling and other functionalities (e.g., VM instantiation, PM startup/shutdown...) typical to real-life Clouds.

In this work, I mostly focus on three components, the energy modelling, the unified resource sharing and the infrastructure simulation, briefly outlined in the following subsections.

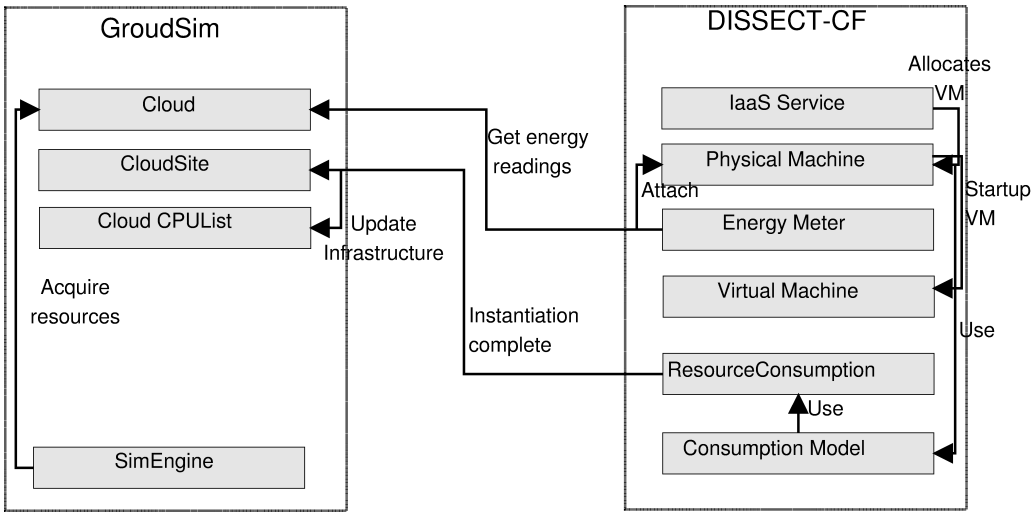


Figure 6.3: Interaction between GroudSim and DISSECT-CF.

6.3.3 DISSECT-CF energy extensions

This section describes the design and implementation of the energy model inside the DISSECT-CF infrastructure simulator [106] and integrated in the user-oriented GroudSim simulator [84]. I provide a closer look to each module in the following sections.

Energy modelling

In this module, I extended the `ConsumptionModel` class with two subclasses (`CPUConsumptionModel` and `LinearConsumptionModel`) to model the instantaneous power consumption consumption according to Equations 2.20 and 2.28.The $\alpha(h)$ and $\beta(h)$ parameters, the idle P_{idle} and maximum P_{max} powers, as well as the $\mathcal{L}(h)$ parameter in Equation 2.20 are set by the user when configuring the simulation. From the combination of these parameters, I get the CPU power consumption $P_{\text{cpu}}(h,t)$. Similarly, the $\gamma(h)$ and $\delta(h)$ parameters in Equation 2.28 define the network and storage power consumptions $P_{\text{net}}(h,t)$ and $P_{\text{io}}(h,t)$. Each class provides an `evaluateConsumption(double load)` method which, when queried, gives the instantaneous power consumption according to the instantaneous load represented by the `load` parameter that models the relative use of the particular resource (e.g. CPU, network,

storage). For CPU, network and I/O load I refer to the equations in the model section, respectively 2.5, 2.6 and 2.7.

Unified resource sharing

In this module, I added support for tasks consuming not only CPU, but also memory resources. This is especially important for live migration, since its time and energy consumption is influenced by the way the memory is used. In [92], I identified the memory dirtying ratio (i.e. the percentage of memory pages marked as dirty over a certain time interval) as one of the most impacting parameters on energy consumption of VM live migration. To support this simulation, I extended the `ResourceConsumption` class to allow the user specify the number of memory pages used by a task and its memory dirtying ratio.If no dirtying ratio is specified during task creation, the task is considered as not modifying its memory pages during execution.

Infrastructure simulation

I based the extension of this module on the studies performed in [92]. To add live VM migration capabilities to DISSECT-CF, both `VirtualMachine` and `PhysicalMachine` classes have been extended.In the `VirtualMachine` class, I added methods allowing the simulation of live migration in six steps.

Initial state transfer The first step of live VM migration is moving the actual state of the VM to the target host without suspending the VM. For this, the hypervisor creates a memory image of the migrating VM and sends it to the target host. I simulate this in multiple steps. First, I allocate a matching resource set for the VM on the target host. If the allocation succeeds, I create a *storage object* of the same VM memory size and simulate its transfer using plain TCP.

Continuous update Since the VM is still running, its state in memory may be modified during the its transfer to the target host. To have a consistent image on the target, I need to transfer these modifications too, requiring

multiple transfers before reaching a consistent state on the target. Xen terminology names each transfer as `pre-copy round` consisting of two steps:

1. *Identify changes in the VM's state and transfer them to the target host.* In real-life, the hypervisor monitors these changes by marking memory pages as dirty if they have been modified by the VM since the initial transfer or the last update. In the simulator, I calculate the amount of dirtied pages called *Written Working Set (WWS)* by using the memory dirtying ratio parameter defined in Section 6.3.3 in the `identifyWWS()` method:

$$WWS(v,t) = \texttt{DR}(v,t) \cdot \Delta t, \tag{6.1}$$

   where d_r is the dirtying ratio (typical amount of pages dirtied in a second) and Δt is the time since the initial transfer or last update. The `identifyWWS()` method returns a storage object of the size of the *WWS*.

2. *Update the state of target host with modifications occurred in the source during state transfer.* Once I identify the *WWS*, I need to transfer it to the target host similarly as in the initial transfer step.

In real-life, the hypervisor performs this phase until a termination criteria is reached (otherwise the migration could run for an indefinite amount of time). In the simulator, I employ the same termination criteria used in Xen's live migration [107]: (1) $WWS \leq 256$ or (2) the maximum number of pre-copy rounds is reached. The maximum number of pre-copy rounds can be either a constant value or dependent on configuration parameters. In Xen, the amount of pre-copy rounds is dependent on the bandwidth levels set inside the configuration. This is because the main focus of VM migration implementation inside Xen hypervisor is to save network bandwidth. To this end, Xen users set inside its configuration files a set of bandwidth levels that are used for VM migration. In the first round, the hypervisor starts the transfer using the lowest bandwidth level set inside the configuration file. Then, if $WWS \geq 256$, the next bandwidth level is used in the following round, until either this condition is reached or the maximum bandwidth level is reached. I

implemented the same algorithm in the simulator, as validation is performed by using data collected measuring Xen hypervisor. In my implementation, it is possible for the user to set a new termination criterion by extending the `VirtualMachine` class and overriding the `migrateLive` method accordingly.

Suspend the VM Once the pre-copy phase is terminated, the VM is suspended to allow the transfer of its last state modifications to the target host. To achieve this, the simulation extension suspends each task running on the VM by using the `suspend` method, and then sends the modifications performed since the last pre-copy round to the target host, similar to the pre-copy phase.

Resume the VM on the target If no error occurred until this point, the state of the VM on the target host is coherent with the source host. Therefore, I resume the tasks suspended in the previous state so that the VM is effectively running on the target host.

Destroy the VM on source After the new VM is resumed, it is possible to release the resources previously owned by the VM on the source host using the `destroy()` method on the instance of the VM running on the source host.

6.3.4 GroudSim/DISSECT-CF

I display in Figure 6.3 how to obtain DISSECT-CF energy readings in GroudSim. To measure the energy consumption in a data centre, I need two basic information: the PMs and their load. For this reason, this operation is performed by the *IaaS Service* of DISSECT-CF responsible for both instantiating the data centre infrastructure and allocating a VM to a suitable PMs. For this purpose, the IaaS Service meter attaches to each host defined in the data centre an `EnergyMeter`. For each host, I define a `ConsumptionModel` for CPU, network and storage that defines the instantaneous power consumption of each component, as discussed in Section 6.3.3. Energy meters collect these instantaneous power consumption values with a user defined frequency and

calculate the energy consumption based on the simulated time spent since the last power measurement. At the end of the simulation, the IaaS Service's meter aggregates the energy consumption for the entire data centre.

6.4 Evaluation

In this section, I evaluate my simulated energy model for VM migrations by first describing the selected benchmarks and the experimental testbed. Then, I describe how I simulate the execution of these benchmarks on the top of GroudSim/DISSECT-CF. Finally, I compare the results of the simulations with energy traces collected from real executions in the simulated Cloud. For benchmarking the implementation of the model inside the simulator, I employed the benchmarks I defined in Section 3.5 and published in [92]. I made this choice because: (1) they already proved their effectiveness in testing the VM migration model, and (2) they allow to check the accuracy of CPU, network and storage models by varying the CPU load and the dirtying ratio, which are the parameters that mostly affect the VM migration.

6.4.1 Benchmarking results

The measurements collected from the benchmarks' execution are shown in Figure 6.4 for the (`m01`, `m02`) machine set, and in Figure 6.5 for the (`o1`, `o2`) machine set. In each chart, the measurements performed in this phase are referred with a label starting with "Real" to distinguish it from the DISSECT-CF results that are the output of the simulation (see the following sections). I show them together with the simulation results for brevity reasons. In each chart, I also distinguish between "-SRC" and "-TRG" measurements taken on the source and target hosts. Each measurement point is traced every 500m according to the power measurement devices' resolution. I average each power reading over ten benchmark runs to ensure statistical significance. For non-live traces, I show the consecutive execution of CPULOAD-SOURCE and CPULOAD-TARGET benchmarks. Concerning live migration, I show the consecutive execution of CPULOAD-

Host	*MAE-NONLIVE*		*NRMSE-NONLIVE*		*MAE-LIVE*		*NRMSE-LIVE*	
	Power [W]	*Energy* [J]	*Power* [%]	*Energy* [%]	*Power* [W]	*Energy* [J]	*Power* [%]	*Energy* [%]
`m01`	42.97	4292.9	16.6	16.9	38.45	5345.8	15.5	8.1
`m02`	51.97	4179.5	18.3	15.1	67.33	6341.8	22	9.3
`o1`	11.98	2248.6	8.2	14.6	27.6	3375.5	18.4	11
`o2`	18.19	2417	14.6	13.2	48.1	5518	14.2	25.6

Table 6.2: Error for the selected machine sets.

SOURCE, MEMLOAD-VM, CPULOAD-TARGET, MEMLOAD-SOURCE, and MEMLOAD-TARGET.

6.4.2 Simulation validation

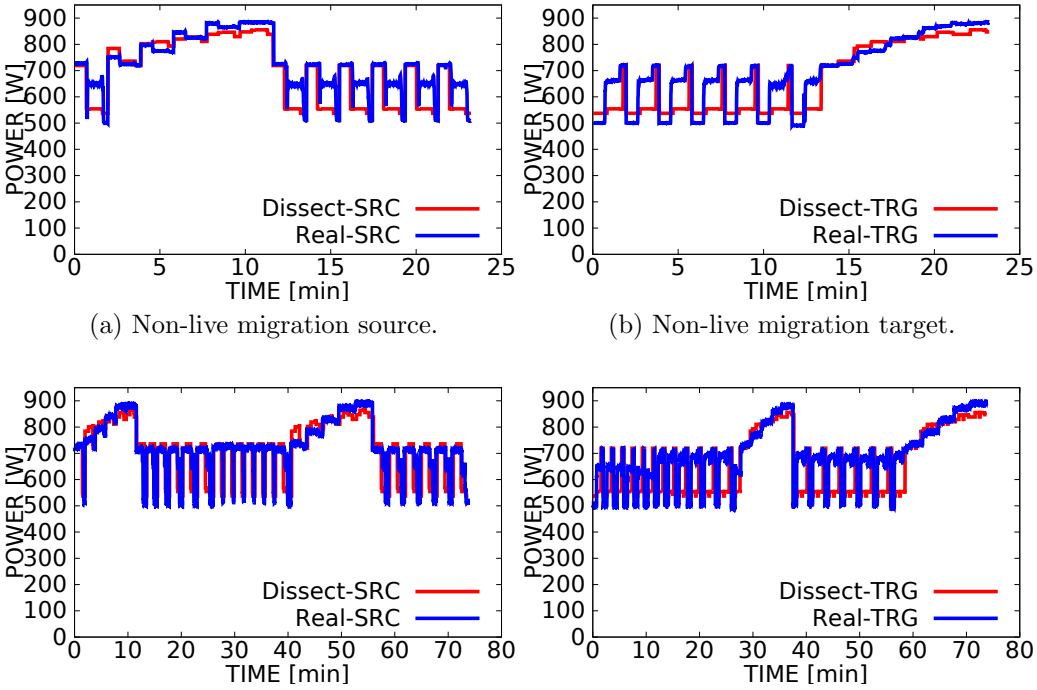
After collecting the real world traces, I implement the same benchmarks on the top of DISSECT-CF to evaluate the accuracy of my simulations. I configure a micro data centre with two PMs matching the configuration of the two kinds of machines I used during the regression modelling (see in Section 6.2). I simulate the load by deploying VMs on the PMs and configure each VM to have 4G of memory to resemble the configuration I used to build my traces. To simulate the execution of the benchmarks, I assign to each VM computing tasks resembling the execution of the selected workloads. I simulate the execution of `matrixmult` by assigning to the VMs a computing task with full CPU utilisation and a dirtying rate of 0.05, as measured over the execution of the real benchmark. In this case, the increasing load on the host is simulated by increasing the number of VMs allocated to the PMs. Concerning `pagedirtier`, I also set the full CPU utilisation, but I vary the dirtying rate for each experimental run as indicated in Table 3.5b. For the non-live migration, I only vary the CPU load since varying the dirtying ratio has an influence on live migration only.

6.4.3 Simulation results

In this section, I compare the results obtained by the simulator with the traces obtained from my experiments. I compute the MAE and NRMSE error metrics on both instantaneous power and energy consumption on both sets of machines. The results are summarised in Table 6.2 for both the machine sets. I also compare in Figure 6.4 (for the (`m01`, `m02`) set) and Figure 6.5 (for the (`o1`, `o2`) set) the power traces obtained from the simulator with the real measurements. I observe that the simulation is able to provide power values with a MAE not higher than 67.3W compared to the real ones. This value is, however, influenced by the fact that in some cases the power consumption is underestimated by around 100W like in Figure 6.4a (between 12 and 24min) and Figure 6.4b (between 0 and 14min) because the test scenarios active during those minutes perform non-live migrations while both hosts are idle In these situations, the simulator only considers the power consumption caused by the network and storage despite some inherent CPU consumption caused by these two operations. Thus, the simulator considers idle CPU consumption for both hosts, despite slight CPU load caused by the need for supporting the storage operations. This slight load, on the other hand, leads to significant offsets in the power consumption model according to the Figure 2.3. In future work, I aim at modelling this inherent CPU load in a generic way to increase the accuracy of the simulator in these unlikely test scenarios too. Nevertheless, NRMSE is in each case between 8% and 22% for instantaneous power consumption, and between 8% and 25% for energy consumption showing that the simulator, by employing my extentions, is able to predict both energy and instantaneous power with good accuracy.

6.4.4 CloudSim comparison

In this section, I compare my simulations with the CloudSim [33] state-of-the-art simulator because it is by far the most widely cited and used in the distributed systems community. CloudSim employs a piecewise model [108] for power consumption, ensuring a fair comparison to my work and allowing simulations of different kinds of architectures.



(a) Non-live migration source.

(b) Non-live migration target.

(c) Live migration source.

(d) Live migration target.

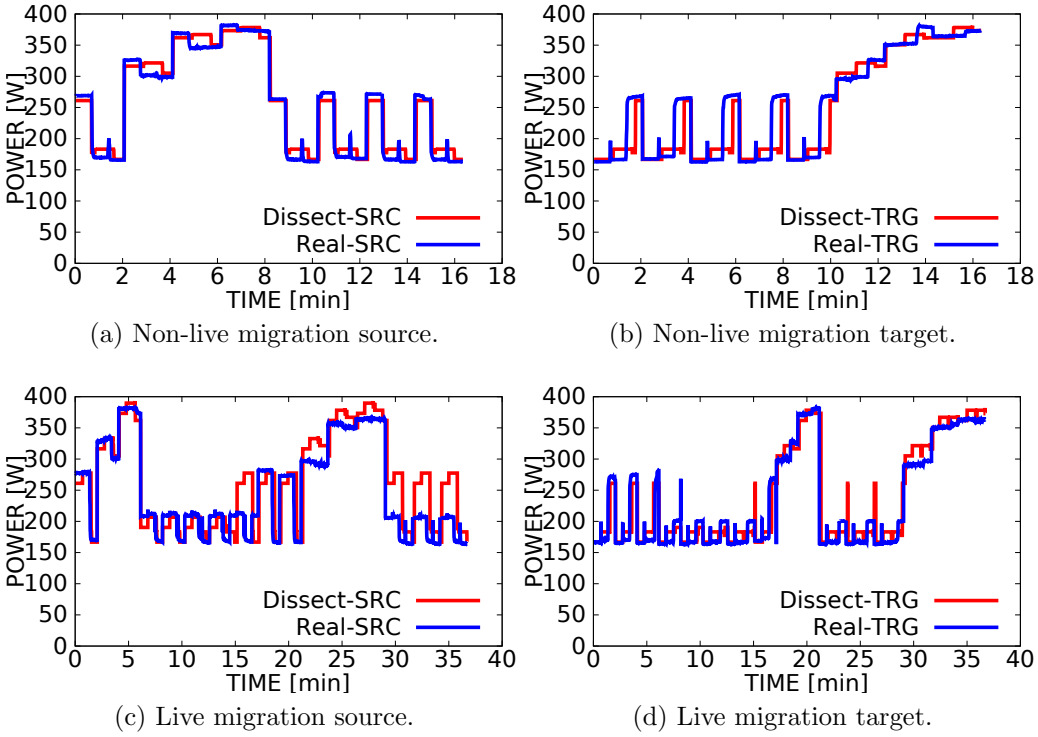
Figure 6.4: Results for the (`m01`, `m02`) machine set.

I did not include DCSim [81] in my evaluation since it does not provide VM migration support at the time of writing required for a fair comparison. I further compared with simulators that use a piecewise linear model for energy consumption only (i.e. CloudSim), since I showed in Section 6.2.1 that linear and cubic models are highly inaccurate for my testbed. For this reason, I did not select SimGrid [32] despite its support for live migration [31] since it uses a linear model for energy consumption [2] Finally, I did not select GreenCloud [80] because it employs a cubic model.

Experimental setup

To perform my comparison, I implemented in CloudSim the same benchmarks used for validating my model in Section 3.5. I initialized a CloudSim

[2] `http://simgrid.gforge.inria.fr/simgrid/3.12/doc/group__SURF__plugin__energy.html#ga166ef80adc810f0990d13539ddfd8adc`



(a) Non-live migration source. (b) Non-live migration target.

(c) Live migration source. (d) Live migration target.

Figure 6.5: Results for the (o1, o2) machine set.

simulation with a data centre consisting of two PMs with the same specifications as in Table 3.5c.

Energy modelling CloudSim allows the user to employ a piecewise linear energy model. For each type of CPU used in the simulation, the user needs to specify an array p with 11 power values, corresponding to the power consumption at different CPU utilisation level, starting from idle to 100% in steps of 10. According to the CPU utilisation level that I have in the simulator, for CloudSim the power consumption of a PM h at a given time instance is given by:

$$P_{cloudsim}(h,t) = p\left[\lceil load_{\text{cpu}}(h,t)\cdot 10\rceil\right] + \delta\cdot load_{\text{cpu}}(h,t) - \\ - \frac{\lfloor load_{\text{cpu}}(h,t)\rfloor}{10}\cdot 100, \quad (6.2)$$

where: $\delta = \frac{p[\lfloor \text{load}_{\text{cpu}}(h,t) \cdot 10 \rfloor] - p[\lceil \text{load}_{\text{cpu}}(h,t) \cdot 10 \rceil]}{10}$ and $\text{load}_{\text{cpu}}(h,t)$ is the CPU utilisation of host h at time instance t as in Equation 2.20. This model is implemented in CloudSim by the `PowerSpecModelPower` class which I extended to model the two different CPU types used.

Benchmark execution In CloudSim, applications running on VMs are modelled by instances of the `CloudLet` object for which it is possible to specify the amount of CPU and memory used. I simulated each benchmark by a `CloudLet` running on VMs with the same number of CPUs and memory as in the real-world experiments. For the CPULOAD benchmarks, I set the `UtilizationModelFull` to simulate full CPU utilisation. I could not simulate the dirtying ratio of the MEMLOAD benchmarks because CloudSim does not allow setting this parameter in the memory utilisation. Nevertheless, I set `UtilizationModelFull` as the memory utilisation model to be sure that the CloudSim implementations of the benchmarks are resembling the real execution as much as possible. Each `CloudLet` uses all resources allocated to the VMs. I simulated the load on the PMs by deploying the corresponding number of VMs, as I do for the real benchmarks (see Chapter 3).

VM migration I simulated VM migration in CloudSim by extending first the `VMSelectionPolicy` class to make sure that only one VM is selected for migration, as I do in the benchmarks. Afterwards, I extended the class `VMAllocationPolicy` to ensure that the selected VM is migrated to the desired target. The VM migration is issued in CloudSim at regular intervals determined by the `SCHEDULING_INTERVAL` parameter. I set this parameter to 40 that is the average time required by the power consumption of PM to stabilise, as explained in Section 3.5). Finally, I set for each simulation the `SIMULATION_LIMIT` parameter to the average execution time obtained in the real benchmark executions.

Results

I display in Table 6.3 the MAE and NRMSE of the energy values obtained from CloudSim and GroudSim/DISSECT-CF compared with real measure-

Simulator	*Dataset*	*MAE* *live* [kWh]	*NRMSE* *live* [%]
GroudSim/ DISSECT-CF	(`m01`, `m02`)	0.0025	9
	(`o1`, `o2`)	0.0024	25.6
CloudSim	(`m01`, `m02`)	0.0190	45.9
	(`o1`, `o2`)	0.0075	68

Table 6.3: GroudSim/DISSECT-CF vs CloudSim comparison.

ments. Since the energy simulation in CloudSim refers to the entire data centre, I compare its prediction with the sum of the energy consumption of the hosts. I only compare the results of the benchmarks that use live migration since CloudSim does not support the non-live one. I observe that using my simulation in GroudSim/DISSECT-CF reduces the NRMSE compared to CloudSim by 36.9% on the (`m01`, `m02`) machine set (i.e. 45.9% versus 9%), and by 42.4% on the (`o1`, `o2`) machine set (i.e. 68% versus 25.6%). I also notice that GroudSim/DISSECT-CF has a lower MAE for both data sets, showing a lower variance compared to CloudSim. I further observe that most errors in CloudSim are due to underestimation. Therefore, the better results are explained not only by the improved accuracy in the CPU model, but also by considering the energy consumption of subsystems ignored by CloudSim such as networking and storage, and the energy overhead of VM migration and the dirtying ratio of each workload. Considering this parameter in GroudSim/DISSECT-CF brings a significant improvement in the accuracy of prediction as shown in [92] because migrating a VM with high dirtying ratio can even double the migration time, which has a high impact on energy consumption.

6.5 Summary

In this chapter, I have shown the implementation details of my model and its validation, comparing it with real measurements and a state-of-the-art Cloud simulator like CloudSim. The results shown in this phase support the fact that current state-of-the-art simulator do not provide an accurate

model for VM migration. Due to the energy consumption of this activity and its importance for VM consolidation, I conclude that more accurate energy modelling for this activity is needed. I propose then a model for energy consumption of VM migration taking into account different power phases, as well as all the involved actors and different hardware components, showing that with my model I am able to increase the accuracy that current models provide.

Chapter 7

Conclusion and future work

7.1 Contributions

In this section I summarise the final contribution of this work, according to the different research areas touched during this work.

7.1.1 Energy model

In Chapter 2 I described the theoretical model and all the foundational aspects behind my work. First, I provided a brief introduction of what is a data centre in my scenario and then discussed virtualization and all the issues that are related to it. Afterwards, I defined what is a VM and introduced a model of the system load. Afterwards, a big part of this chapter focused on VM migration and (1) the VM migration approaches on which the thesis focuses, (2) the actors involved in the migration process and (3) the definition of the energy phases identified during VM migration. Afterwards, I described the energy model of each data centre component, more precisely CPU, network and storage. On top of these models, I built a model for VM migration. The model consists of two parts: the energy model and the runtime model. The energy model is defined for each one of the energy phases that I previously identified, and related to the resource utilisation of the main actors involved in the VM migration. Afterwards, I defined a model for VM migration runtime.

7.1.2 Network modelling

In Chapter 4, my research efforts concerning the modelling of energy consumption of network transfers are discussed. First, I introduced NNETS, a versatile network benchmarking tool offering eight configuration parameters, some not covered by existing tools (e.g. variable traffic patterns, full duplex connections). Second, I designed a set of benchmarks and evaluated the energy efficiency of the NICs' software stacks in different configurations covering a wide spectrum of possible application behaviours. Third, I introduced energy models capable of providing accurate estimations based on the NIC type of adapter and transfer characteristics including payload size, connection concurrency and traffic patterns with an average error of 6.1%. Fourth, I tested the accuracy of my model in predicting energy consumption of a non-live VM migration process, obtaining an average error of 9.8%. Fifth, I proposed a set of guidelines for choosing the most energy efficient NIC.

7.1.3 VM migration modelling

In Chapter 5, I validated a new energy model for VM migration. I considered the impact of workloads running on different actors and identified how much their load impacts the energy consumption of VM migration. Then, I compared the accuracy of my model versus other state-of-the-art models that do not consider it. I quantified how much each actor's workload influences VM migration energy-wise. My results demonstrated an improvement up to 24% in accuracy, showing that the workload impact on VM migration cannot be ignored when predicting its energy consumption. As a result, employing my model can significantly improve energy efficiency through VM consolidation. For example, one may think not to consolidate a VM with an high dirtying ratio to a PM that is running many VMs running CPU intensive workloads since this increases the energy consumption of VM migration. The other models considered in this work do not take into account impact of workloads running on the target host and therefore, may not be able to provide the same accuracy in predictions. Such a model could also be easily

integrated in Cloud simulators to provide more accurate estimation of energy consumption in data centres. I plan to extend this work by also considering the impact of network and storage-intensive workloads.

7.1.4 Integration into simulation

In Chapter 6, I evaluated my model's accuracy using traces collected from two different types of PMs in a private Cloud, showing a relative error lower than 18% on both data sets. Afterwards, I implemented the model in the user-side GroudSim simulator by exploiting its integration with DISSECT-CF infrastructure simulator. I evaluated the accuracy of my implementation by comparing it with real measurements, showing a NRMSE between 8% and 22% for power prediction and between 8% and 25.6% for energy estimation. Finally, I compared the results obtained by my implementation with the CloudSim state-of-the-art simulator showing an improvement of at least 36.8% in energy prediction accuracy. In the future, I plan to perform further extensions to the simulator by improving the energy models for network and storage and use them for studying the effects of different energy-aware consolidation algorithms in modern virtualized data centres. I am further interested in validating the simulator with different real-world benchmarks such as TPC-C[1] or SPECPower[2].

7.2 Future work

7.2.1 Virtualization

Regardless of the modelling of PMs, there are several functionalities introduced by the virtualization layer that need to be considered to improve existing simulators. There are different types of virtualization, but in this section I only focus on the ones that are more of interest for Cloud computing: OS-level virtualization and container-based virtualization. Then, I consider the

[1] `http://www.tpc.org/tpcc/`
[2] `https://www.spec.org/benchmarks.html#power`

capability of performing overcommitment, which is widely used in virtualized data centres to increase resource utilization, thus reducing energy consumption. Finally, I focus on the migration feature, offered by many modern hypervisors that allows to dynamically reallocate load inside the data centre and to take more energy efficient decisions.

Resource overcommitment

Several papers proposed models for data centre energy consumptions, however, they either focus on a specific CPU architecture [35] or assume a linear relationship between CPU usage and energy consumption [56], which may lead to inaccurate results [40]. VM migration's [107] energy consumption has been modelled in many different works [109, 110], but none them considers over-commitment and the impact of different data centre actors on energy consumption as done by [92], and have not been implemented in a simulator. Resource overcommitment consists of allocating to VMs more resources than available on the PMs. The main advantage of using this technique is that utilisation of PMs, and consequently their energy efficiency, is increased. To ensure that each VM gets a fair share of the resources, smart resource scheduling mechanisms are needed, such as the credit scheduler in Xen [86]. Models for resource overcommitment have been proposed by [111, 112]. However, in existing simulators like CloudSim [33] the only possibility to perform overcommitment is to manually specify the maximum amount of resources that each VM gets. To further improve accuracy, simulators should implement such resource allocation algorithms and provide an accurate simulation of their effects on energy consumption and VMs performance.

VM migration

VM migration's [107] energy consumption has been modelled in many different works [109, 110]. Efforts for implementing VM migration models in simulation have been made by CloudSim [33] and SimGrid [113] but they either (1) consider only non-live migration, (2) do not consider effects on performance of the VM migration or (3) do not consider energy consumption

of the VM migration. Some works like [92] consider overcommitment and the impact of different data centre actors on energy consumption, but at the moment of writing no implementation in a Cloud simulator exists. Due to its impact on data centre performance and energy consumption, implementing such models in Cloud simulators will for sure increase accuracy of existing simulations.

Container-based virtualization

Container-based virtualization has recently emerged in modern data centres, thanks to technologies like Docker[3], OpenVZ[4] and LXC[5]. In this type of virtualization, the guest OS runs as application inside the OS. This reduces the overhead introduced by the hypervisor and eliminates the additional software layers between OS and hardware. Recent works started investigating the advantages of using containers in place of VMs. In [114] an extensive performance comparison of VMs and containers is performed. In [115] further analysis of the advantages of using containers instead of VMs is performed. In [116] Docker containers are used for workflow execution on Clouds, showing their applicability to scientific applications. According to these preliminary results, containers seem to be a promising alternative to VMs for high-performance computing in the Cloud, since they provide virtualization with a lower overhead compared to the typical OS-based virtualization. However, at the moment no simulator supports this virtualization technique. Providing to the distributed systems community the possibility to simulate the container-based virtualization technology would enable scientists to choose which kind of virtualization better suits their applications.

7.3 Discussion

According to my analysis, none of the existing simulators is able to capture all the aspects of data centre behaviour. For example, GreenCloud is more

[3] `http://www.docker.com`
[4] `http://www.openvz.org`
[5] `http://www.linuxcontainers.org`

focused on network simulations, but it does not provide an accurate support to the virtualization infrastructure, especially concerning VM migration and hypervisor's behaviour. Moreover, it does not provide support for overcommitment at the time of writing. According to my analysis, the simulators providing more features are CloudSim and SimGrid, but they still miss many important ones. For example, CloudSim provides energy modelling, but it does not provide the same flexibility as SimGrid in defining applications. Moreover, there are many aspects that are often not considered, such as (1) different hypervisors' behaviour, (2) resource overcommitment and (3) memory behavior, including swapping. All these aspects must be considered for an accurate energy modelling. Moreover, at the time of writing, there is no data centre simulator that is considering the possibility to have container-based virtualization. Container-based virtualization is an alternative to the typical VM-based virtualization, where Linux containers are used instead of VMs. At the moment there is no simulator for running container-based virtualization. Providing the capability of accurately simulating this type of virtualization would give the possibility to assess the best type of virtualization (VM or container-based) for different application scenarios. Finally, very little attention is paid to validation of the simulators. Therefore, in the future I plan to focus my research in the following areas:

- Improving accuracy of physical modelling;
- Support of different types of virtualization, including container-based virtualization;
- Providing more detailed models of overcommitment;
- Improving accuracy of VM migration modelling;
- Providing the user with a general validation framework.

List of Figures

List of Tables

Bibliography

[1] Gabor Kecskemeti. Dissect-cf: A simulator to foster energy-aware scheduling in infrastructure clouds. *Simulation Modelling Practice and Theory*, 58, Part 2:188 – 218, 2015. Special issue on Cloud Simulation.

[2] L.A. Barroso and U. Holzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, Dec 2007.

[3] Chandra Chekuri and Sanjeev Khanna. On multidimensional packing problems. *SIAM J. Comput.*, 33(4):837–851, Apr 2004.

[4] V. Shrivastava, P. Zerfos, K. w. Lee, H. Jamjoom, Y. H. Liu, and S. Banerjee. Application-aware virtual machine migration in data centers. In *INFOCOM '11*, pages 66–70. IEEE, 2011.

[5] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang. Joint vm placement and routing for data center traffic engineering. In *INFOCOM '12*, pages 2876–2880. IEEE, 2012.

[6] Y. Guo, A. L. Stolyar, and A. Walid. Shadow-routing based dynamic algorithms for virtual machine placement in a network cloud. In *INFOCOM '13*, pages 620–628. IEEE, 2013.

[7] Z. Xiao, W. Song, and Q. Chen. Dynamic resource allocation using virtual machines for cloud computing environment. *Transactions on Parallel and Distributed Systems*, 24(6):1107–1117, 2013.

[8] H. Xu and B. Li. Anchor: A versatile and efficient framework for resource management in the cloud. *Transactions on Parallel and Distributed Systems*, 24(6):1066–1076, 2013.

[9] Weiwei Fang, Xiangmin Liang, Shengxin Li, Luca Chiaraviglio, and Naixue Xiong. Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers. *Computer Networks*, 57(1):179–196, Jan 2013.

[10] M. Alicherry and T. V. Lakshman. Optimizing data access latencies in cloud systems by intelligent virtual machine placement. In *INFOCOM '13*, pages 647–655. IEEE, 2013.

[11] K. Zheng, X. Wang, L. Li, and X. Wang. Joint power optimization of data center network and servers with correlation analysis. In *INFOCOM '14*, pages 2598–2606. IEEE, 2014.

[12] L. Chen and H. Shen. Consolidating complementary vms with spatial/temporal-awareness in cloud datacenters. In *INFOCOM '14*, pages 1033–1041, 2014.

[13] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *NSDI'05*, pages 273–286. USENIX.

[14] A Strunk and W. Dargie. Does live migration of virtual machines cost energy? In *AINA '13*, pages 514–521. IEEE, 2013.

[15] K. Rybina, W. Dargie, A. Strunk, and A. Schill. Investigation into the energy cost of live migration of virtual machines. In *SustainIT '13*, pages 1–8. IEEE, Oct 2013.

[16] Qiang Huang, Fengqian Gao, Rui Wang, and Zhengwei Qi. Power consumption of virtual machine live migration in clouds. In *CMC'11*, pages 122–125. IEEE, 2011.

[17] Haikun Liu, Cheng-Zhong Xu, Hai Jin, Jiayu Gong, and Xiaofei Liao. Performance and energy modeling for live migration of virtual machines. In *HPDC '11*, pages 171–182. ACM, 2011.

[18] Anja Strunk. A lightweight model for estimating energy cost of live migration of virtual machines. In *CLOUD '13*, pages 510–517. IEEE, June 2013.

[19] Qiang Huang, Fengqian Gao, Rui Wang, and Zhengwei Qi. Power consumption of virtual machine live migration in clouds. In *CMC '11*, pages 122–125. IEEE, April 2011.

[20] N. Bobroff, A. Kochut, and K. Beaty. Dynamic placement of virtual machines for managing sla violations. In *INM '07*, pages 119–128. IFIP/IEEE, 2007.

[21] X. Meng, V. Pappas, and L. Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *INFOCOM '10*, pages 1–9. IEEE, 2010.

[22] H. Liu and B. He. Vmbuddies: Coordinating live migration of multi-tier applications in cloud environments. *Transactions on Parallel and Distributed Systems*, 26(4):1192–1205, 2015.

[23] Dennis Abts, Michael R. Marty, Philip M. Wells, Peter Klausler, and Hong Liu. Energy proportional datacenter networks. *SIGARCH Comput. Archit. News*, 38(3):338–347, Jun 2010.

[24] Lei Huang, Qin Jia, Xin Wang, Shuang Yang, and Baochun Li. Pcube: Improving power efficiency in data center networks. In *CLOUD '11*, pages 65–72. IEEE, 2011.

[25] Dzmitry Kliazovich, Pascal Bouvry, and Samee Ullah Khan. Dens: Data center energy-efficient network-aware scheduling. In *GREENCOM-CPSCOM '10*, pages 69–75. IEEE, 2010.

[26] Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In *NSDI '08*, pages 323–336. USENIX, 2008.

[27] Hang-Sheng Wang, Li-Shiuan Peh, and S. Malik. A power model for routers: modeling alpha 21364 and infiniband routers. In *High Performance Interconnects, 2002. Proceedings. 10th Symposium on*, pages 21–27. IEEE, 2002.

[28] P. Alonso, R.M. Badia, J. Labarta, M. Barreda, M.F. Dolz, R. Mayo, E.S. Quintana-Orti, and R. Reyes. Tools for power-energy modelling and analysis of parallel scientific applications. In *ICPP '12*, pages 420–429, 2012.

[29] Barry Rountree, David K. Lowenthal, Shelby Funk, Vincent W. Freeh, Bronis R. de Supinski, and Martin Schulz. Bounding energy consumption in large-scale mpi programs. SC '07, pages 49:1–49:9. ACM, 2007.

[30] Siddhartha Jana, Oscar Hernandez, Stephen Poole, and Barbara Chapman. Power consumption due to data movement in distributed programming models. In *Euro-Par '14*, volume 8632, pages 366–378. Springer International Publishing, 2014.

[31] T. Hirofuchi, A. Lebre, and L. Pouilloux. Adding a live migration model into simgrid: One more step toward the simulation of infrastructure-as-a-service concerns. In *5th International Conference on Cloud Computing Technology and Science*, volume 1, pages 96–103, 2013.

[32] Henri Casanova, Arnaud Legrand, and Martin Quinson. Simgrid: A generic framework for large-scale distributed experiments. In *P2P '09*, pages 126–131. IEEE, 2009.

[33] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, and Rajkumar Buyya. Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, January 2011.

[34] Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. The cost of a cloud: Research problems in data center networks. *SIGCOMM Comput. Commun. Rev.*, 39(1):68–73, December 2008.

[35] Yiyu Chen, Amitayu Das, Wubi Qin, Anand Sivasubramaniam, Qian Wang, and Natarajan Gautam. Managing server energy and operational costs in hosting centers. *SIGMETRICS Perform. Eval. Rev.*, 33(1):303–314, Jun 2005.

[36] P. Gschwandtner, M. Knobloch, B. Mohr, D. Pleiter, and T. Fahringer. Modeling cpu energy consumption of hpc applications on the ibm power7. In *Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on*, pages 536–543, Feb 2014.

[37] Yakun Sophia Shao and David Brooks. Energy characterization and instruction-level energy model of intel's xeon phi processor. In *Proceedings of the 2013 International Symposium on Low Power Electronics and Design*, ISLPED '13, pages 389–394, Piscataway, NJ, USA, 2013. IEEE Press.

[38] James Hamilton. Internet-scale service infrastructure efficiency. *SIGARCH Comput. Archit. News*, 37(3):232–232, June 2009.

[39] Suzanne Rivoire, Parthasarathy Ranganathan, and Christos Kozyrakis. A comparison of high-level full-system power models. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, HotPower'08, pages 3–3, Berkeley, CA, USA, 2008. USENIX Association.

[40] A.-C. Orgerie, L. Lefevre, and J.-P. Gelas. Demystifying energy consumption in grids and clouds. In *Green Computing Conference, 2010 International*, pages 335–342, 2010.

[41] David Wang, Brinda Ganesh, Nuengwong Tuaycharoen, Kathleen Baynes, Aamer Jaleel, and Bruce Jacob. Dramsim: A memory system simulator. *SIGARCH Comput. Archit. News*, 33(4):100–107, November 2005.

[42] Y. Kim, W. Yang, and O. Mutlu. Ramulator: A fast and extensible dram simulator. *Computer Architecture Letters*, PP(99):1–1, 2015.

[43] Carl A. Waldspurger. Memory resource management in vmware esx server. *SIGOPS Oper. Syst. Rev.*, 36(SI):181–194, December 2002.

[44] Guilin Zhang, Huiqiang Wang, L.V. Hongwu, Guangsheng Feng, and Zhanbo He. A dynamic memory management model on xen virtual machine. In *Mechatronic Sciences, Electric Engineering and Computer (MEC), Proceedings 2013 International Conference on*, pages 1609–1613, 2013.

[45] Irfan Habib. Virtualization with kvm. *Linux J.*, 2008(166), February 2008.

[46] Qingbo Zhu, Francis M. David, Christo F. Devaraj, Zhenmin Li, Yuanyuan Zhou, and Pei Cao. Reducing energy consumption of disk storage using power-aware cache management. In *The 10th International Conference on High-Performance Computer Architecture (HPCA-10)*, pages 118–129, 2004.

[47] David P. Helmbold, Darrell D. E. Long, and Bruce Sherrod. A dynamic disk spin-down technique for mobile computing. In *Proceedings of the 2Nd Annual International Conference on Mobile Computing and Networking*, MobiCom '96, pages 130–142, New York, NY, USA, 1996. ACM.

[48] D. Colarelli and D. Grunwald. Massive arrays of idle disks for storage archives. In *Supercomputing, ACM/IEEE 2002 Conference*, pages 47–47, 2002.

[49] P.M. Greenawalt. Modeling power management for hard disks. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 1994., MASCOTS '94., Proceedings of the Second International Workshop on*, pages 62–66, 1994.

[50] Reem Alshahrani and Hassan Peyravi. Modeling and simulation of data center networks. In *Proceedings of the 2Nd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM PADS '14, pages 75–82, New York, NY, USA, 2014. ACM.

[51] Nongda Hu, Binzhang Fu, Xiufeng Sui, Long Li, Tao Li, and Lixin Zhang. Dcnsim: A unified and cross-layer computer architecture simulation framework for data center network research. In *Proceedings of the ACM International Conference on Computing Frontiers*, CF '13, pages 19:1–19:9, New York, NY, USA, 2013. ACM.

[52] Hiroki Shirayanagi, Hiroshi Yamada, and Kenji Kono. Honeyguide: A vm migration-aware network topology for saving energy consumption in data center networks. *2014 IEEE Symposium on Computers and Communications (ISCC)*, 0:000460–000467, 2012.

[53] Yueping Zhang, Ao-Jan Su, and Guofei Jiang. Evaluating the impact of data center network architectures on application performance in virtualized environments. In *Quality of Service (IWQoS), 2010 18th International Workshop on*, pages 1–5, June 2010.

[54] Vincenzo De Maio, Vlad Nae, and Radu Prodan. Evaluating energy efficiency of gigabit ethernet and infiniband software stacks in data centres. In *Proceedings of the 7th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2014)*. IEEE Computer Society, 2014.

[55] A.-C. Orgerie, L. Lefevre, I. Guerin-Lassous, and D.M.L. Pacheco. Ecofen: An end-to-end energy cost model and simulator for evaluating power consumption in large-scale networks. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pages 1–6, 2011.

[56] Steven Pelley, David Meisner, Thomas F Wenisch, and James W VanGilder. Understanding and abstracting total data center power. *Workshop on Energy-Efficient Design*, 2009.

[57] Toni Mastelic, Ariel Oleksiak, Holger Claussen, Ivona Brandic, Jean-Marc Pierson, and Athanasios V. Vasilakos. Cloud computing: Survey on energy efficiency. *ACM Comput. Surv.*, 47(2):33:1–33:36, December 2014.

[58] Aman Kansal, Feng Zhao, Jie Liu, Nupur Kothari, and Arka A. Bhattacharya. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pages 39–50, New York, NY, USA, 2010. ACM.

[59] Mansoor Alicherry and T. V. Lakshman. Network aware resource allocation in distributed clouds. In *INFOCOM '12*, pages 963–971. IEEE, 2012.

[60] Brandon Heller, Srinivasan Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown. Elastictree: Saving energy in data center networks. In *NSDI '10*, pages 249–264. USENIX, 2010.

[61] Yunfei Shang, Dan Li, and Mingwei Xu. Energy-aware routing in data center network. In *SIGCOMM '10 workshop on Green networking*, pages 1–8. ACM, 2010.

[62] Haijin Yan, Scott A. Watterson, David K. Lowenthal, Kang Li, Rupa Krishnan, and Larry L. Peterson. Client-centered, energy-efficient wireless communication on ieee 802.11b networks. *Transactions on Mobile Computing*, 5(11):1575–1590, 2006.

[63] Maruti Gupta and Suresh Singh. Greening of the internet. In *SIGCOMM '03*, pages 19–26. ACM, 2003.

[64] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright. Power awareness in network design and routing. In *INFOCOM '08*, pages 457–465. IEEE, 2008.

[65] Q. Wang, M. Hempstead, and W. Yang. A realistic power consumption model for wireless sensor network devices. In *SAHCN '06*, volume 1, pages 286–295, 2006.

[66] L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *INFOCOM '01*, volume 3, pages 1548–1557, 2001.

[67] Jayant Baliga, Robert Ayre, Kerry Hinton, Wayne V. Sorin, and Rodney S. Tucker. Energy consumption in optical ip networks. *J. Lightwave Technol.*, 27(13):2391–2403, Jul 2009.

[68] J. Chan and S. Parameswaran. Nocee: energy macro-model extraction methodology for network on chip routers. In *ICCAD '05*, pages 254 – 259. ACM, 2005.

[69] A.-C. Orgerie, L. Lefevre, I. Guerin-Lassous, and D.M.L. Pacheco. Ecofen: An end-to-end energy cost model and simulator for evaluating power consumption in large-scale networks. In *WoWMoM*, pages 1–6. IEEE, 2011.

[70] Wei Deng, Hai Jin, Xiaofei Liao, Fangming Liu, Li Chen, and Haikun Liu. Lifetime or energy: Consolidating servers with reliability control in virtualized cloud datacenters. In *CloudCom '12*, pages 18–25. IEEE Computer Society, 2012.

[71] Jen-Cheng Huang, Hsien-Hsin S. Lee, and Mohammad M. Hossain. Migration energy-aware workload consolidation in enterprise clouds. In *CloudCom '12*, pages 405–410. IEEE, 2012.

[72] Richa Sinha, Nidhi Purohit, and Hiteishi Diwanji. Energy efficient dynamic integration of thresholds for migration at cloud data centers. *IJCA Special Issue on Communication and Networks*, (1):44–49, Dec 2011.

[73] Anton Beloglazov and Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurr. Comput. : Pract. Exper.*, 24(13):1397–1420, Sep 2012.

[74] William Voorsluys, James Broberg, Srikumar Venugopal, and Rajkumar Buyya. Cost of virtual machine live migration in clouds: A performance evaluation. In *CloudCom '09*, pages 254–265. Springer-Verlag, 2009.

[75] S. Akoush, R. Sohan, A Rice, AW. Moore, and A Hopper. Predicting the performance of virtual machine migration. In *MASCOTS '10*, pages 37–46. IEEE, Aug 2010.

[76] Akshat Verma, Gautam Kumar, Ricardo Koller, and Aritra Sen. Cosmig: Modeling the impact of reconfiguration in a cloud. In *MASCOTS '11*, pages 3–11. IEEE, 2011.

[77] Kateryna Rybina, Abhinandan Patni, and Alexander Schill. Analysing the migration time of live migration of multiple virtual machines. In *CLOSER '14*, pages 590–597. Scitepress, 2014.

[78] W. Dargie. Estimation of the cost of vm migration. In *ICCCN '14*, pages 1–8, Aug 2014.

[79] A Strunk. Costs of virtual machine live migration: A survey. In *SERVICES '12*, pages 323–329. IEEE, June 2012.

[80] Dzmitry Kliazovich, Pascal Bouvry, and SameeUllah Khan. Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, 62(3):1263–1283, 2012.

[81] Michael Tighe, Gastón Keller, Michael Bauer, and Hanan Lutfiyya. Dcsim: A data centre simulation tool for evaluating dynamic virtualized resource management. In *CNSM '12*, pages 385–392. IEEE, 2012.

[82] Simon Ostermann, Kassian Plankensteiner, and Radu Prodan. Using a new event-based simulation framework for investigating resource provisioning in clouds. *Scientific Programming*, 19(2-3):161–178, 2011.

[83] Simon Ostermann, Kassian Plankensteiner, Radu Prodan, Thomas Fahringer, and Alexandru Iosup. Workflow monitoring and analysis tool for ASKALON. In *CoreGRID '08*, pages 1–14, 2008.

[84] Gabor Kecskemeti, Simon Ostermann, and Radu Prodan. Fostering energy-awareness in simulations behind scientific workflow management systems. In *Proceedings of the 7th IEEE/ACM International Con-*

ference on Utility and Cloud Computing, UCC 2014, London, United Kingdom, December 8-11, 2014, pages 29–38, 2014.

[85] A. Nuñez, J. L. Vázquez-Poletti, A. C. Caminero, J. Carretero, and I. M. Llorente. Design of a new cloud computing simulation platform. In *Proceedings of the 2011 International Conference on Computational Science and Its Applications - Volume Part III*, ICCSA'11, pages 582–593, Berlin, Heidelberg, 2011. Springer-Verlag.

[86] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, 37(5):164–177, October 2003.

[87] Brian Walters. Vmware virtual platform. *Linux J.*, 1999(63es), July 1999.

[88] Michael R. Hines, Umesh Deshpande, and Kartik Gopalan. Post-copy live migration of virtual machines. *SIGOPS Oper. Syst. Rev.*, 43(3):14–26, Jul 2009.

[89] Fabien Hermenier, Xavier Lorca, Jean-Marc Menaud, Gilles Muller, and Julia Lawall. Entropy: A consolidation manager for clusters. In *VEE '09*, pages 41–50. ACM.

[90] Akshat Verma, Puneet Ahuja, and Anindya Neogi. pmapper: power and migration cost aware application placement in virtualized systems. In *Middleware '08*, pages 243–264. Springer-Verlag, 2008.

[91] Aziz. Murtazaev and Sangyoon. Oh. Sercon: Server Consolidation Algorithm using Live Migration of Virtual Machines for Green Computing. volume 28, pages 212–231. Taylor & Francis online, 2011.

[92] Vincenzo De Maio, Gabor Kecskemeti, and Radu Prodan. A workload-aware energy model for virtual machine migration. In *CLUSTER '15*. IEEE, 2015.

[93] Yaniv Ben-Itzhak, Israel Cidon, and Avinoam Kolodny. Performance and power aware cmp thread allocation modeling. In *HiPEAC'10*, pages 232–246. Springer-Verlag, 2010.

[94] The problem of power consumption in servers. `https://software.intel.com/sites/default/files/m/d/4/1/d/8/power_consumption.pdf`. Accessed: 2016-03-06.

[95] Robert Basmadjian, Nasir Ali, Florian Niedermeier, Hermann de Meer, and Giovanni Giuliani. A methodology to predict the power consumption of servers in data centres. In *e-Energy '11*, pages 1–10. ACM, 2011.

[96] Adam Wade Lewis, Nian-Feng Tzeng, and Soumik Ghosh. Runtime energy consumption estimation for server workloads based on chaotic time-series approximation. *Trans. Archit. Code Optim.*, 9(3):15:1–15:26, Oct 2012.

[97] Theophilus Benson, Aditya Akella, and David A. Maltz. Network traffic characteristics of data centers in the wild. In *IMC '10*, pages 267–280. ACM, 2010.

[98] Gong Chen, Wenbo He, Jie Liu, Suman Nath, Leonidas Rigas, Lin Xiao, and Feng Zhao. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *NSDI'08*, pages 337–350. USENIX, 2008.

[99] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le. Rapl: Memory power estimation and capping. In *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*, pages 189–194, 2010.

[100] J. Chu and V. Kashyap. Transmission of IP over InfiniBand (IPoIB). RFC 4391, 2006.

[101] V. Kashyap. IP over InfiniBand: Connected Mode. RFC 4755, 2006.

[102] W. Li, H. Yang, Z. Luan, and D. Qian. Energy prediction for mapreduce workloads. In *DASC '11*, pages 443–448, Dec 2011.

[103] A-C. Orgerie, L. Lefevre, and J.-P. Gelas. Demystifying energy consumption in grids and clouds. In *IGCC '10*, pages 335–342, Aug 2010.

[104] Faraz Ahmad and T. N. Vijaykumar. Joint optimization of idle and cooling power in data centers while maintaining response time. *SIGARCH Comput. Archit. News*, 38(1):243–256, Mar 2010.

[105] Jyothi Sekhar, Getzi Jeba, and S. Durga. A survey on energy efficient server consolidation through vm live migration. *IJAET*, 5:515–525, November 2012.

[106] Gabor Kecskemeti. DISSECT-CF: a simulator to foster energy-aware scheduling in infrastructure clouds. *Simulation Modelling Practice and Theory*, 58P2:188–218, November 2015.

[107] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 273–286, 2005.

[108] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Gener. Comput. Syst.*, 28(5):755–768, May 2012.

[109] Haikun Liu, Hai Jin, Cheng-Zhong Xu, and Xiaofei Liao. Performance and energy modeling for live migration of virtual machines. *Cluster Computing*, 16(2):249–264, 2013.

[110] Anja Strunk. A lightweight model for estimating energy cost of live migration of virtual machines. In *2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA, June 28 - July 3, 2013*, pages 510–517, 2013.

[111] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes. Efficient datacenter resource utilization through cloud resource overcommitment. In *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, pages 330–335, 2015.

[112] Rahul Ghosh and Vijay K. Naik. Biting off safely more than you can chew: Predictive analytics for resource over-commit in iaas cloud. In Rong Chang, editor, *IEEE CLOUD*, pages 25–32. IEEE, 2012.

[113] T. Hirofuchi, A. Lebre, and L. Pouilloux. Adding a live migration model into simgrid: One more step toward the simulation of infrastructure-as-a-service concerns. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 1, pages 96–103, 2013.

[114] Wes Felter, Alexandre Ferreira, Ram Rajamony, and Juan Rubio. An updated performance comparison of virtual machines and linux containers. In *2015 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2015, Philadelphia, PA, USA, March 29-31, 2015*, pages 171–172, 2015.

[115] Stephen Soltesz, Herbert Pötzl, Marc E. Fiuczynski, Andy Bavier, and Larry Peterson. Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors. *SIGOPS Oper. Syst. Rev.*, 41(3):275–287, March 2007.

[116] Wolfgang Gerlach, Wei Tang, Kevin Keegan, Travis Harrison, Andreas Wilke, Jared Bischof, Mark D'Souza, Scott Devoid, Daniel Murphy-Olson, Narayan Desai, and Folker Meyer. Skyport: Container-based execution environment management for multi-cloud scientific workflows. In *Proceedings of the 5th International Workshop on Data-Intensive Computing in the Clouds*, DataCloud '14, pages 25–32. IEEE Press, 2014.