Université de Neuchâtel
Faculté des Sciences
Institut d'Informatique

Multi-Level Energy Efficiency for Heterogeneous Data Centers

par

Mascha Kurpicz-Briki

Thèse

présentée à la Faculté des Sciences
pour l'obtention du grade de Docteur ès Sciences

Acceptée sur proposition du jury:

Prof. Pascal Felber, Directeur de thèse
Université de Neuchâtel, Suisse

Dr. Anita Sobe, Co-Directrice de thèse
Accenture Zürich, Suisse

Dr. Sonia Ben Mokhtar
CNRS, France

Prof. Rüdiger Kapitza
Technische Universität Braunschweig, Allemagne

Prof. Peter Kropf
Université de Neuchâtel, Suisse

Soutenue le 20.09.2016

UNIVERSITÉ DE NEUCHÂTEL

Faculté des sciences
Secrétariat-décanat de Faculté
Rue Emile-Argand 11
2000 Neuchâtel - Suisse
Tél: + 41 (0)32 718 2100
E-mail: secretariat.sciences@unine.ch

IMPRIMATUR POUR THESE DE DOCTORAT

La Faculté des sciences de l'Université de Neuchâtel autorise l'impression de la présente thèse soutenue par

Madame Mascha KURPICZ-BRIKI

Titre:

"Multi-Level Energy Efficiency for Heterogeneous Data Centers"

sur le rapport des membres du jury composé comme suit:

- Prof. Pascal Felber, directeur de thèse, Université de Neuchâtel, Suisse
- Prof. Peter Kropf, Université de Neuchâtel, Suisse
- Dr Sonia Ben Mokhtar, Université Claude Bernard, LIRIS, Lyon, France
- Prof. Rüdiger Kapitza, TU Braunschweig, Allemagne
- Dr Anita Sobe, Accenture, Zürich, Suisse

Neuchâtel, le 25 octobre 2016 Le Doyen, Prof. R. Bshary

Contents

List of Tables

List of Figures

Acknowledgements

I want to thank my advisors, Prof. Pascal Felber and Dr. Anita Sobe, for their precious help over the last few years. I learnt a lot and gained many new skills thanks to you. Furthermore I want to thank Dr. Sonia Ben Mokhtar, Prof. Rüdiger Kapitza and Prof. Peter Kropf for being part of my jury.

I want to thank my colleagues from the IIUN for the technical and non-technical discussions, the coffee breaks, the ice cream breaks with view on the lake, the aperos, the table tennis competitions, the BCN tours and the climbing adventures. I have spent an unforgettable time the last past years, thanks to you all.

Thanks to the HiPEAC framework and ARM Research in Cambridge (UK) I had the amazing chance to gain some industry insights and to collaborate with brilliant people.

I want to thank all my friends for supporting me and forgiving my last-minute dinner cancellations due to paper deadlines. The journey of my thesis would not have been the same without you!

Des Weiteren möchte ich mich bei meiner Familie bedanken: ohne eure Unterstützung und Ermutigung wäre all das, was ich über die letzten Jahre erreicht habe, nicht möglich gewesen.

Tout particulièrement, j'aimerais remercier mon mari pour son soutien quotidien indéfectible et sa joie de vivre qui me permet de réussir dans tout ce que je fais.

Abstract

The ICT sector has an important impact on global energy consumption (building devices, building networks, operation, air-conditioning and more). Studies show that cloud computing as a whole consumes more energy than entire countries like Germany or India. Furthermore, recent estimates have shown that the cloud computing sector, and thus the energy consumed by data centers, is still increasing.
The major goal of this thesis is to increase the energy awareness in heterogeneous data centers and thus contribute to the reduction of energy consumption in the ICT sector. To improve the energy awareness, we need to know how much energy is spent. This is influenced not only by the infrastructure, but by every single application running in the data center. Unfortunately we cannot measure application energy consumption with physical power meters. Therefore, this thesis contains real-world models to estimate the energy consumption at different levels.
This work is organized as a stack with 3 layers: (1) data center, (2) host/virtualization, (3) end user applications. At the data center layer, we first study the impact of different workloads running on heterogeneous machines. In a second step, we develop EPAVE, a model for energy-proportional accounting in VM-based environments. EPAVE is supported by PowerIndex, a profiling and energy estimation framework. At the host/virtualization layer, we present BitWatts, a middleware toolkit for building software-defined power meters. With BitWatts we cross the boundaries of virtual environments and provide an estimation of the power consumption of applications running within virtual machines.
At the bottom of the stack we look into battery modelling to extend the battery life of mobile devices.

Keywords: virtualized systems, energy estimation, power estimation, energy awareness, heterogeneous environments, accounting, battery modelling

Résumé

Le secteur des TIC (Technologies de l'Information et de la Communication) est grand consommateur d'énergie (assemblage d'appareils, constructions de réseaux, opérations, refroidissement et autres). Des études montrent que l'ensemble des activités de l'informatique en nuage consomme plus d'énergie que des pays entiers comme l'Allemagne ou l'Inde. De plus, des estimations récentes montrent que le secteur de l'informatique en nuage, et donc l'énergie consommée par les centres de données sur lesquels elle repose, est encore en croissance.

Dans cette thèse, je développe des approches pour faciliter la consommation énergétique responsable des centres de données hétérogénes et par conséquent réduire la consommation d'énergie globale du secteur des TIC. Pour faciliter la consommation énergétique responsable, il faut savoir combien d'énergie est consommée. Ceci est influencé par l'infrastructure, mais aussi par chaque application qui tourne dans le centre de données. Malheureusement, on ne peut pas appliquer un wattmètre à une application. Pour contourner cette limitation, cette thèse propose des modèles réalistes pour estimer la consommation énergétique à plusieurs niveaux.

Cette thèse est organisée comme une pile à 3 niveaux: (1) centre de données, (2) virtualisation, (3) applications clientes. Au premier niveau des centres de données nous commençons par une première phase d'étude de l'influence de l'hétérogénéité matérielle sur la consommation de puissance des applications. Dans une deuxième phase, nous développons EPAVE, un modèle pour la facturation proportionnelle à l'énergie dans les environnements virtualisés. EPAVE est supporté par PowerIndex, un framework pour le profilage et l'estimation d'énergie. Au deuxième niveau de la virtualisation nous présentons BitWatts, un outil middleware pour construire des wattmètres logiciels. Avec BitWatts, on franchit la barrière des environnements virtualisés pour proposer une estimation de consommation pour des applications s'exécutant au sein de machines virtuelles. Enfin, au troisième niveau, nous modélisons les batteries d'appareils mobiles et proposons une approche pour prolonger leur durée de vie.

Mots clefs: systèmes virtualisés, estimation d'énergie, estimation de puissance, consommation énergétique responsable, environnements hétérogènes, modélisation de batterie

Zusammenfassung

Der ICT Sektor hat einen erheblichen Einfluss auf den globalen Energieverbrauch (Herstellung von Geräten, Netzwerke, Betrieb, Kühlung und so weiter). Studien zeigen, dass die Cloud insgesamt mehr Energie verbraucht als ganze Länder wie Deutschland oder Indien. Des Weiteren belegen kürzlich erstellte Schätzungen, dass der Cloud Sektor weiter wächst, und somit der Energieverbrauch weiter steigen wird.
Das Ziel dieser Dissertation ist es, das Energiebewusstsein in Datenzentren zu erhöhen und somit zur Senkung des Energieverbrauchs im ICT Sektor beizutragen. Das Bewusstsein zu mehr Energieeffizienz wird gesteigert, wenn man weiss, wie viel tatsächlich an Energie verbraucht wird. Einfluss darauf haben nicht nur die Infrastruktur alleine, sondern auch jede einzelne Applikation, die im Datenzentrum ausgeführt wird. Leider kann man an eine Applikation kein Wattmeter anschliessen. Deshalb befasst sich diese Dissertation mit realitätsnahmen Modellen um den Energieverbrauch auf mehreren Ebenen abzuschätzen.
Diese Dissertation ist als Stack mit 3 Ebenen aufgebaut: (1) Datenzentrum, (2) Host/Virtualisierung, (3) Anwenderapplikationen. Auf der Ebene der Datenzentren analysieren wir zuerst den Einfluss von heterogenen Maschinen auf den Stromverbrauch von Applikationen. In einer zweiten Phase entwickeln wir EPAVE, ein Model für energieproportionales Verrechnen in virtualisierten Umgebungen. EPAVE wird von PowerIndex unterstützt, einem Framework für Profiling und Energieschätzung. Auf der Ebene der Virtualisierung stellen wir BitWatts vor, ein Middleware Toolkit um software-basierte Strommessgeräte zu bauen. Mit BitWatts überschreiten wir die Grenze zu virtuellen Umgebungen und schätzen den Stromverbrauch von Applikationen, die in virtuellen Maschinen laufen. Auf der untersten Ebene des Stacks widmen wir uns Batteriemodellierung, mit dem Ziel, das Batterieleben von mobilen Geräten zu verlängern.

Stichwörter: virtualisierte Systeme, Energieschätzung, Stromverbrauchschätzung, Energiebewusstsein, heterogene Umgebungen, Verrechnungssysteme, Batteriemodellierung

List of Publications

Kurpicz-Briki, M., S. Diestelhorst, And P. Felber.
Impact of Mobile Applications on Battery Life: Power Characterization and Mitigation Measures.
Technical report, outcome of a HiPEAC internship grant with ARM Research in Cambridge, UK.

Kurpicz, M., A-C. Orgerie, A. Sobe, and P. Felber.
Energy-proportional Profiling and Accounting in Heterogeneous Virtualized Environments.
Has been submitted to the Elsevier Special Issue Energy-Sim for Sustainable Computing Journal.

Kurpicz, M., A-C. Orgerie, and A. Sobe.
How much does a VM cost? Energy-proportional Accounting in VM-based Environments.
24th International Conference on Parallel, Distributed and Network-Based Processing (PDP 2016), 02/2016.

Palomar, O., S. Rethinagiri, G. Yalcin, R. Titos-Gil, P. Prieto, E. Torrella, O. Unsal, A. Cristal, P. Felber, A. Sobe, Y. Hayduk, M. Kurpicz et al.
Energy Minimization at All Layers of the Data Center: The ParaDIME Project.
2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2016.

Colmant, M., M. Kurpicz, P. Felber, L. Huertas, R. Rouvoy, and A. Sobe.
Process-level Power Estimation in VM-based Systems.
EuroSYS, Bordeaux, France, ACM, 04/2015.

Colmant, M., M. Kurpicz, P. Felber, L. Huertas, R. Rouvoy, and A. Sobe.
BitWatts: A Process-level Power Monitoring Middleware.
Middleware 2014 Poster/Demos, Bordeaux, France, ACM, 12/2014.

Kurpicz, M., A. Sobe, and P. Felber.
Using Power Measurements as a Basis for Workload Placement in Heterogeneous Multi-Cloud Environments.
CrossCloudBrokers '14 (co-located to Middleware 2014), Bordeaux, France, ACM, 12/2014.

Kurpicz, M., M. Colmant, L. Huertas, A. Sobe, P. Felber et R. Rouvoy.
Est-ce que ton application dans le nuage est économe en énergie? Estimation de la puissance par processus dans des systèmes virtualisés.
ComPAS'2014 poster session, Neuchâtel, Switzerland, 04/2014.

1 Introduction

1.1 Problem Statement

With the paradigm shift towards cloud computing, more and more data centers appear all around the world. Even though newer generations of hardware are getting more energy-efficient, today's cloud computing requires more electricity in the form of energy than entire countries such as India or Germany [1]. Figure 1.1 shows the electricity consumption of different countries and cloud computing in the year 2007 as reported by Greenpeace [1].

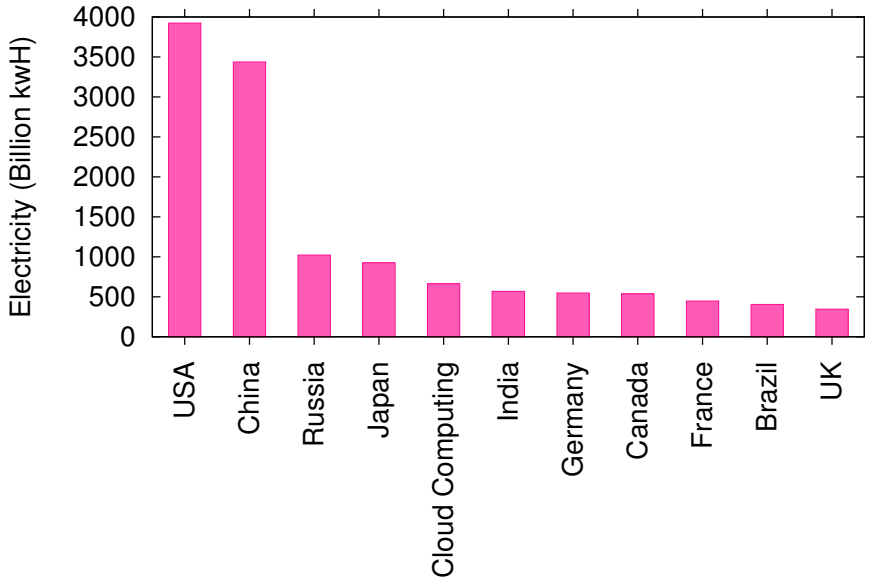

Figure 1.1: Cloud computing consumed more electricity than countries like India or Germany (from Greenpeace [1]) in 2007.

Even though these numbers are from the year 2007, we know that the cloud computing sector, and thus the energy consumed by cloud computing, is still increasing [2]. Recent estimates show that the electricity consumption of data centers will increase from 15% of the total electricity consumption of the ICT sector (as of 2012) to 21%

by 2017. Figure 1.2 shows the increase of the electricity consumption of data centers and depicts that the electricity consumption for personal devices will decrease.

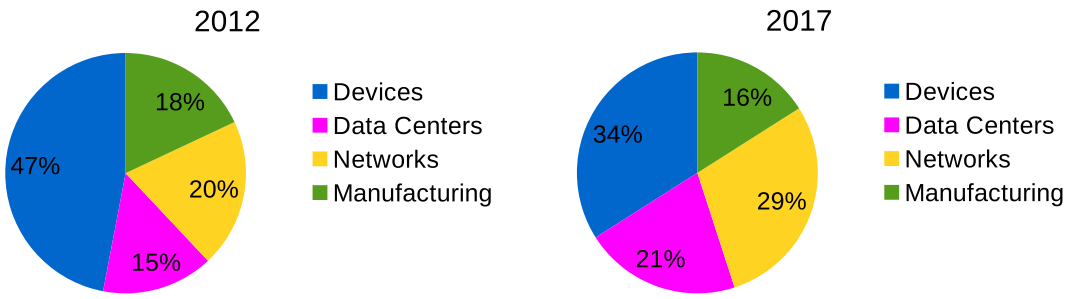

Figure 1.2: Main components of electricity consumption for the ICT sector, 2012 vs. 2017 (from [2]).

A more recent Greenpeace report [3] states that in 2015, 8% of the global electricity consumption is caused by the ICT sector, which is more than 2000 Terawatt-hours per year.

In many state-of-the-art works, the authors assume that data centers comprise servers with the same hardware of the same type. However, this is not realistic, as we often find different generations of the same hardware or even different types of hardware in real-world data centers [4]. The heterogeneity of data centers has an impact on the energy efficiency and we must consider this diversity when studying or estimating energy consumption.

The major goal of this thesis is to increase the energy awareness in heterogeneous data centers and thus contribute to the reduction of energy consumption in the ICT sector. To reduce the overall energy consumption in data centers, we need to have knowledge of the energy consumption of different components. Knowing the energy consumption of hardware with physical power meters is not enough. Abstract concepts like virtualization and distributed systems create more complexity. Furthermore, it is often not realistic (in terms of effort and costs) to instrument entire data centers with physical power meters to measure the energy directly at the hardware. We therefore need ways to *estimate* the energy consumption.

In this thesis I provide energy estimation at different abstraction layers. Each layer of the stack contributes to the overall energy consumed by a system. To get an understanding of the energy consumption of the entire system, we need to estimate the energy for each layer.

- Data center
- Host/virtualization
- End user applications

Data centers are the top layer of the stack. To achieve energy awareness in heterogeneous data centers, we need a prediction on how much energy a workload will consume on heterogeneous machines. This knowledge can then be applied to different tasks as energy-efficient scheduling or energy-proportional pricing models. Therefore, I want to answer the following research questions for data centers:

Is there a relevant change in the power footprint of an application when it runs on heterogeneous hardware, and how can we take advantage of this for workload placement?

How can we provide transparent, reproducible and predictive cost calculation in heterogeneous data centers based on the energy consumption?

Host and virtualization are placed at the second layer of the stack. When it comes to energy consumption, settings made by the operating system are critical. For example, the Linux operating system can control the CPU frequency of the machine using *governors*.[1] To achieve overall energy awareness, we need to have fine-grained knowledge of the energy consumption at a process-level. A particular challenge are virtualized operating systems, which add an additional layer of abstraction.
In this thesis I want to answer the following research question at the host and virtualization layer:

Which processes, on the host and in virtual machines, are consuming the most power?

At the bottom of the stack we consider end user applications. Batteries are being used nowadays to power a wealth of different devices, ranging from tiny sensors to smartphones and even electrical cars. Saving energy and thus improving the battery life is a crucial concern. At the same time, modern CPUs and associated co-processors are becoming more and more powerful and energy demanding. They include multiple cores with various consumption characteristics, and the drained power is highly dependent on the configuration used at a given point in time. Battery lifetime can

[1] https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt

Hardware	Model	Cores/Threads	RAM (GB)	TDP (W)
m_1: Intel Xeon	2x X5365 (3.0GHz)	2x 4/4	5	2x 150
m_2: AMD	4x Opteron 8354 (2.2GHz)	4x 4/4	8	4x 115

Table 1.1: Hardware characteristics of machines m_1 and m_2.

suffer variations depending on these characteristics. The research question for this layer of the stack is as follows:

How can we take advantage of dynamic power consumption characteristics to extend the battery life of mobile devices?

In this thesis I raise the energy awareness problem and provide approaches for fine-grained energy estimation at different levels of abstraction.

1.2 Motivation

The domain of power and energy estimation is a very challenging area of research. Apart from the multiple layers discussed in the previous section, many factors impact the actual power consumption of a workload execution.
Let's consider a very simple example where we execute a benchmark (*Bonnie++*[2]) on two different machines m_1 and m_2. Bonnie++ is a disk benchmark with different phases, including sequential output, sequential input and random seeks. In the phase of sequential output it writes single characters and entire blocks, and it modifies blocks. For the sequential input it reads character by character and entire blocks. In the end it comprises a phase of random seeks where the benchmark measures the physical movements of the head of the hard disk. We execute the benchmark on two different machines with hardware as shown in Table 1.1. Figure 1.3 shows the power consumption of the two machines during the benchmark execution, measured with a physical power meter.
We notice that the range of power values is already different between the two machines. This is due to the different *idle power*. As idle power we consider the power that a machine consumes for being switched on, without running any workload. The power traces show clearly the separate phases of the benchmark. However, we see that there is a notable difference between the power behavior on machine m_1 and

[2]http://sourceforge.net/projects/bonnie, accessed on 17.07.2016

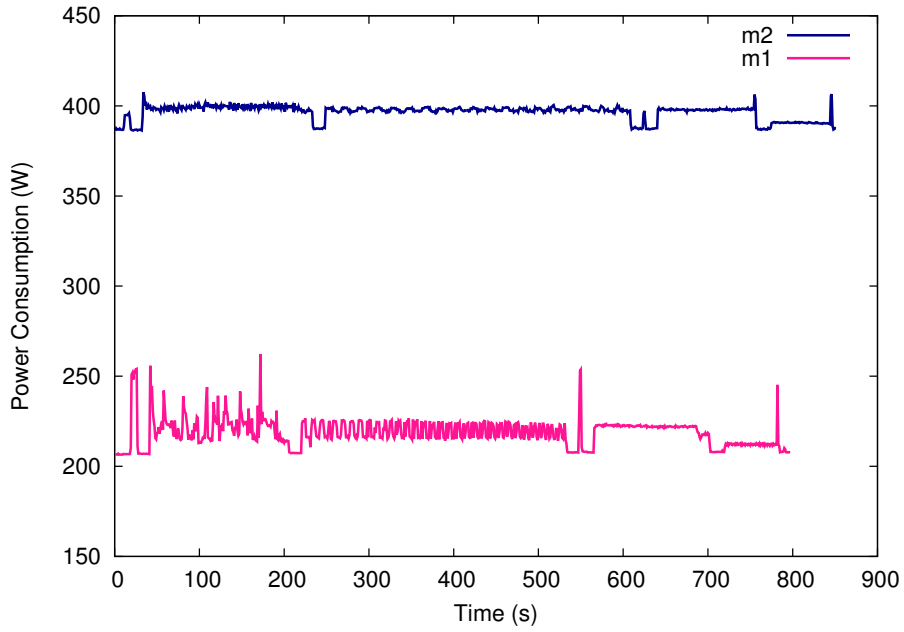

Figure 1.3: Power trace of the Bonnie++ workload on machines m_1 and m_2.

m_2. Also, the execution time for the same workload differs slightly between the two machines.

We note that even in this simple setup where we execute the very same workload on two machines, it has a different power footprint on each system. Depending on the age of the machine, the hardware configuration and, particularly in this case, the state of the hard disk (fragmentation, file system, *etc.*), the power consumption is different. These are all factors that we need to consider whenever we want to make power-efficient decisions, estimate the power consumption or simply compare the power consumption of two systems. Additional factors such as hardware features and temperature are presented in more detail in the next chapter.

1.3 Contribution

As presented previously, the contributions of this thesis are presented as a stack.

For data centers I studied the impact of heterogeneity on several workloads. Based on this we developed energy-proportional price models in EPAVE [5] and extended it with PowerIndex [6], an energy estimation framework.

At the second layer of the stack, I address the operating system layer with BitWatts [7], a toolkit for process-level power estimation on physical machines and in

virtualized environments. BitWatts allows the operating system to understand which processes are the hotspots of power consumption.
For end user applications, and in collaboration with ARM Research in Cambridge (UK),[3] I tried to extend the battery life of recent smartphones.
The contributions of this thesis address the research questions as follows:

Is there a relevant change in the power footprint of an application when it runs on heterogeneous hardware, and how can we take advantage of this for workload placement? In the first contribution of this thesis, we study the performance and energy efficiency of a set of heterogeneous architectures for multiple micro-benchmarks (stressing CPU, memory and disk) and for a real-world cloud application. We observe from our results that some architectures are more energy-efficient for disk-intensive workloads, whereas others are better for CPU-intensive workloads. This study provides the basis for workload characterization and cross-cloud scheduling under constraints of energy efficiency. This work was presented as a paper *Using Power Measurements as a Basis for Workload Placement in Heterogeneous Multi-Cloud Environments* at the CrossCloudBrokers Workshop 2014 at the ACM/IFIP/USENIX Middleware Conference in Bordeaux, France.

How can we provide transparent, reproducible and predictive cost calculation in heterogeneous data centers based on the energy consumption? As a second contribution we introduce EPAVE, a model for Energy-Proportional Accounting in VM-based Environments. EPAVE allows transparent, reproducible and predictive cost calculation for users and for cloud providers. It comes with PowerIndex, a profiling and estimation model, which is able to profile the energy costs of a VM on a given server architecture and can then estimate its energy costs on a different one. Preliminary results of this work were published in the 24th International Conference on Parallel, Distributed and Network-Based Processing (PDP 2016). An extended version with the title *Energy-proportional Profiling and Accounting in Heterogeneous Virtualized Environments* has been submitted to a special issue Energy-Sim for Sustainable Computing published by Elsevier.

Which processes, on the host and in virtual machines, are consuming the most power? The third contribution presented in this thesis is BitWatts. BitWatts is a middleware toolkit for building software-defined power meters. Such software meters provide an accurate alternative to dedicated hardware systems or embedded

[3] HiPEAC internship October 2015 – January 2016

power counters by estimating power consumption in the small, i.e., at the level of software processes. With BitWatts we cross the boundaries of virtual environments and provide an estimation of the power consumption of applications running within virtual machines (VMs). Preliminary results were presented at the Conférence en Parallélisme, Architecture et Système (ComPAS) 2014 in Neuchâtel, Switzerland and at the ACM/IFIP/USENIX Middleware Conference 2014 in Bordeaux, France at the poster sessions. The paper *Process-Level Power Estimation in VM-based Systems* was then published and presented in 2015 at EuroSYS in Bordeaux, France.

How can we take advantage of dynamic power consumption characteristics to extend the battery life of mobile devices? In this work, we study the impact of dynamic characteristics on battery life by collecting and analyzing real power traces. We particularly focus on uneven patterns that can adversely affect the battery life, and we propose a potential solution to mitigate their effect. This contribution is a result of a HiPEAC internship grant with ARM Research in Cambridge, where I stayed 4 months as a PhD intern.

2 Background and Related Work

As mentioned before, many factors influence the power consumption. In this chapter we will look into *how* the different factors impact the power consumption. In particular, we will consider hardware features such as HyperThreading, DVFS and turbo features. Whenever a machine executes heavy calculations, the temperature of the CPU may rise. Therefore, this chapter also covers background information about temperature.
Furthermore, we present the hardware, define metrics and discuss the related work. The work presented in this thesis has been performed in collaboration with the ParaDIME project [8]. Background information about the project is given in this chapter as well.

2.1 Hardware

Table 2.1 gives an overview of the architecture characteristics of a variety of systems that comprise either AMD CPUs or Intel CPUs. In the evaluation of the chapters of this thesis, different subsets of the listed machines were used. Even though they are typically not used in data centers, we also consider mobile CPUs (i7 and VIA) as they are currently among the most energy-efficient architectures, typically running on battery power.
The selected systems include multicore CPUs or multiprocessors with frequencies between 1.6GHz and 3.4GHz. The Xeon (from 2007) has two processors with an overall of TDP of 300 W. The AMD (from 2008) with four processors and a total TDP of 460 W can also be power hungry. While the AMD, the AMD-TC and the Xeon have a high number of cores (8 to 16 cores), some of the Intel machines (i3, i5, i7, Haswell) have a lower core number but support HyperThreading.
The selected machines do not only differ by their CPU, but also by their hard disks. Most of the hard disks rotate 7200 times per minute (RPM), only the i5 and the mobile devices (i7 and VIA) have 5900 RPM and 5400 RPM respectively. The cache

Nr.	Hardware	Model	Cores/ Threads	RAM (GB)	TDP (W)	HDD	Size	RPM	Cache
1	Intel i3	i3-2100 (3.1GHz)	2/4	6	65	WDC	250GB	7200	16MB
2	Intel i5	i5-650 (3.2GHz)	2/4	4	73	WDC	1TB	5900	64MB
3	Intel i7	i7-2620M (2.7GHz)	2/4	4	35	WDC	750GB	5400	8MB
4	Intel Xeon	2x X5365 (3.0GHz)	2x 4/4	5	2x 150	Seagate	250GB	7200	8MB
5	Intel Haswell	i7-4770 (3.4GHz)	4/8	12	84	Seagate	2TB	7200	64MB
6	AMD	4x Opteron 8354 (2.2GHz)	4x 4/4	8	4x 115	Hitachi	500GB	7200	16MB
7	AMD-TC	FX-8120 (3.1GHz)	8/8	8	125	Seagate	1TB	7200	32MB
8	VIA	C7-M ULV (1.6GHz)	1/1	2	8	TOSHIBA	120GB	5400	8MB
9	Intel Xeon	e5-2630 (2.3GHz)	2x 6/12	32	2x 95	-	-	-	-
10	AMD Opteron	250 (2.4GHz)	2/2	2	215	-	-	-	-
11	Intel Xeon	W3520 (2.66GHz)	4/8	-	130	-	-	-	-

Table 2.1: Hardware characteristics of the selected systems.

size generally depends on the size of the hard disk, i.e, larger disks comprise a larger cache.

2.2 Metrics

2.2.1 Power vs. Energy

The *power* is an instantaneous metric and is measured in Watt. The *energy* is defined as the power over a certain period of time. For example, if the power consumption is constant, we can compute the energy E as the median power P multiplied by the execution time T in seconds:

$$E = P * T$$

If we want to compare the execution of two workloads on two different machines, we must consider the energy and not the power. It would not be fair to compare the power consumption, since the execution time can be different.

To illustrate this, consider the following example. We run a factorial computation on two different machines: m_3 and m_4. Figure 2.1 depicts the power consumption of the computation on both machines. The power consumption of m_3 is approximately 72W and thus lower than the power consumption of m_4 (around 80W). However, the execution time to complete the computation is longer on m_3. Therefore, we require more energy to execute the computation on m_3. It costs 3390J to run on m_3 and only 1810J to run on m_4.

Given this difference, we must use carefully power or energy depending on the context and depending on what we want to compare or measure.

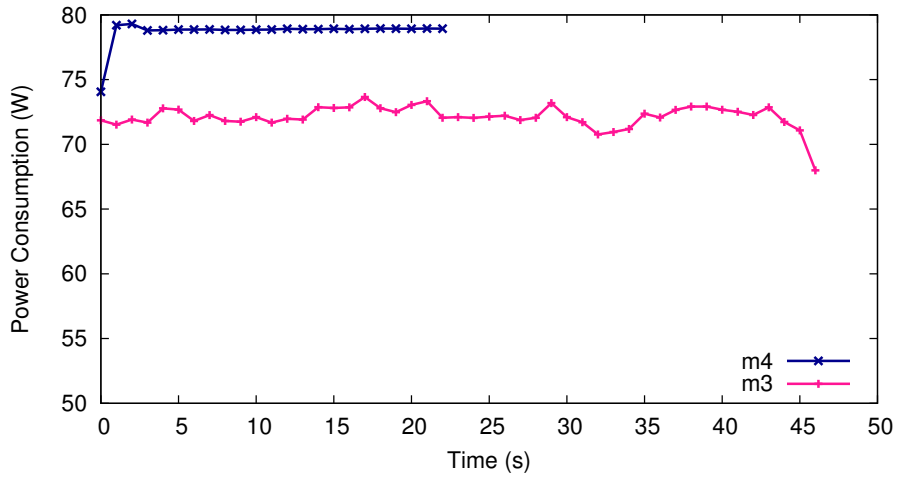

Figure 2.1: The power consumption of m_3 is lower, but the execution time is longer, and thus, we consume more energy for the calculation.

2.2.2 Physical Power Meters

Different approaches exist to measure the power consumption of single machines or in data centers. Power can be measured at different levels, for example for an entire rack of servers, for a single machine or within the components of a machine (e.g. only the processor).

In most works presented in this thesis we used a Alciom PowerSpy power meter.[1] This wattmeter is attached to the power plug of a machine (or other devices) and sends in real-time power data using Bluetooth. The power meter comes with Windows drivers, but an open-source driver for Linux was developed in the context of this thesis.[2]

We use the values obtained from the physical power meter as a ground truth for our power estimation approaches. Furthermore, we can use the measured values to study the impact of different components on the power consumption.

[1] http://www.alciom.com/en/products/powerspy2-en-gb-2.html, accessed on 12.07.2016
[2] https://github.com/patrickmarlier/powerspy.py

2.2.3 Thermal Design Power

The *Thermal Design Power (TDP)* for a specific hardware can typically be extracted from the manufacturer's data sheet. It corresponds to the average maximum power a processor can dissipate while running a program. This is not the maximum power that a processor can reach. There can be moments in the execution where the power is higher than the TDP, but then temperature will raise. When the temperature hits the limit, the processor can scale down the frequency of the calculations.

2.3 Impact of Hardware Features on the Power Consumption

This section describes different factors that have an impact on the power consumption of a machine. For a CPU-intensive workload, for example, the temperature may rise with time. Also, different hardware features can impact the power consumption. Older machines tend to have simpler architectures and thus are less complex to understand in terms of power behavior.
In particular, in this thesis, I consider the following hardware features:

HyperThreading. HyperThreading technology enables multiple threads to share the same physical core.[3] It allows us to run applications simultaneously but still maintaining the system responsiveness.

Dynamic Voltage and Frequency Scaling (DVFS). DVFS is a power management technique on modern computer architectures [9]. The frequency (and thus voltage) of the CPU cores is dynamically adapted, depending on current applications.

TurboBoost/TurboCORE. Another feature provided by recent computer architectures is TurboCORE (AMD)[4] / TurboBoost (Intel).[5] These features allow CPUs to temporarily run over the normally allowed operation frequency. This is done via dynamic control of the processor's clock rate and it is activated whenever the operating system asks for the highest performance state.

[3] http://www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html, accessed on 12.07.2016
[4] http://www.amd.com/en-us/innovations/software-technologies/turbo-core, accessed on 12.07.2016
[5] http://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html, accessed 12.07.2016

As a first example we consider a very simple AMD machine with 48 cores. It does have no HyperThreading, no DVFS and no turbo feature. We execute the *stress* command to generate CPU load, we load a different number of cores (*i.e .,* 6,12,18,24,30,36,42,48) and measure the power each time. Figure 2.2 shows the median power consumption for each configuration.

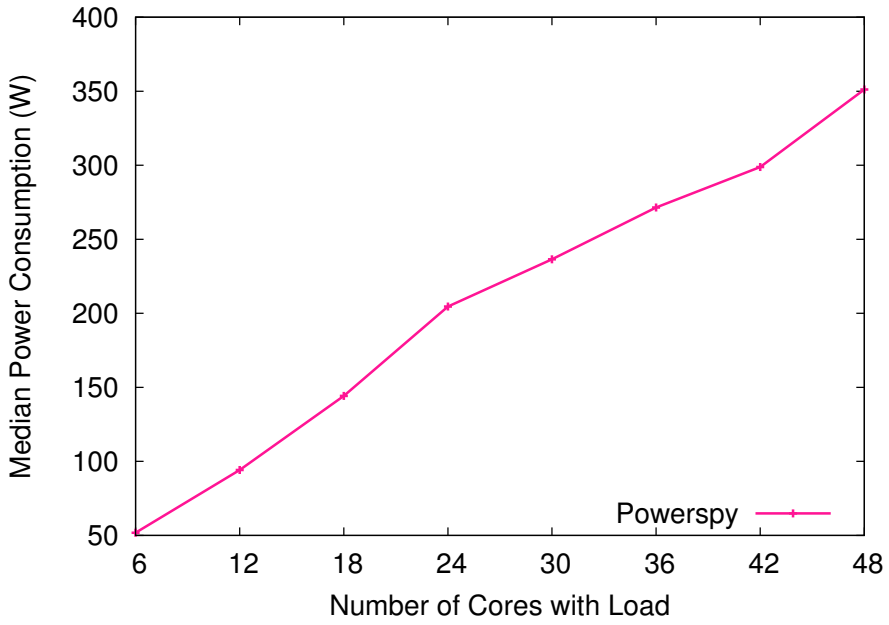

Figure 2.2: Median power consumption for load on an increasing number of cores.

We notice that the curve is almost linear. The curve is slightly higher between 18 and 42 cores being stressed. This machine has four processors with 12 cores each. By default, the processes will be scheduled across the processors to obtain a load balancing for reliability. For example, when we stress four cores, the processes will be load-balanced over all four processors. Each processor will have load on one core. To make the power curve perfectly linear, we can pin the stress processes manually to the cores. When we pin the stress processes to cores, we first fill all available cores on the first processor. Then we start loading the second processor, and so on. Figure 2.3 shows the results of the same experiment, with manual assignment of processes to cores.

We notice that now the power increase is perfectly linear with increasing number of cores being stressed. This is due to the simple architecture of the machine, without particular hardware features.

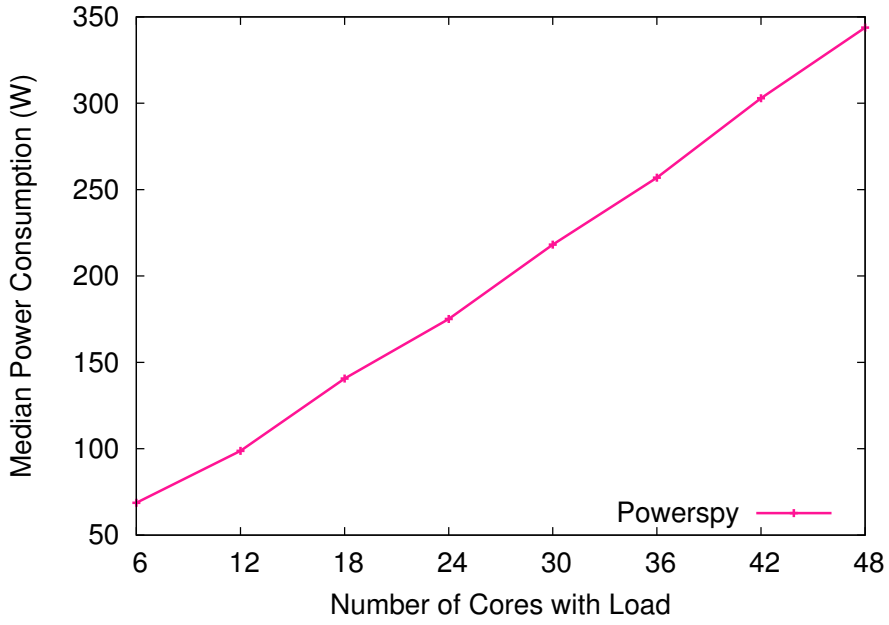

Figure 2.3: Median power consumption for load on an increasing number of cores (pinned).

2.3.1 Impact of DVFS and Turbo features

In this subsection we will look into the impact of DVFS and turbo features on the power consumption. We look at the power values of an 8-core AMD FX-8120 machine with a regular frequency of 1400MHz to 3100MHz and a turbo frequency that reaches 3400MHz on more than 4 cores and 4000MHz on 4 cores or less. Using the Linux *userspace governor*, we manually set the CPU frequency to the available frequencies. Figure 2.4 shows the median power consumption for load on 2 and 4 cores. Load was generated using the Linux *stress* utility.
The power consumption is lower for CPU load on only two cores. In both cases, the power consumption increases with a higher frequency. In this example we only reach 3100MHz, we are not able to set the turbo frequencies manually using this utility.

2.3.2 Impact of HyperThreading

Another hardware feature that has a large impact on the power behavior is Hyper-Threading.
To show the impact of this technology on the power consumption, we consider an Intel i3 machine with 2 physical cores and 4 hardware threads implementing Hyper-Threading. We stress 1 to 4 cores using the Linux *stress* utility and compute for each

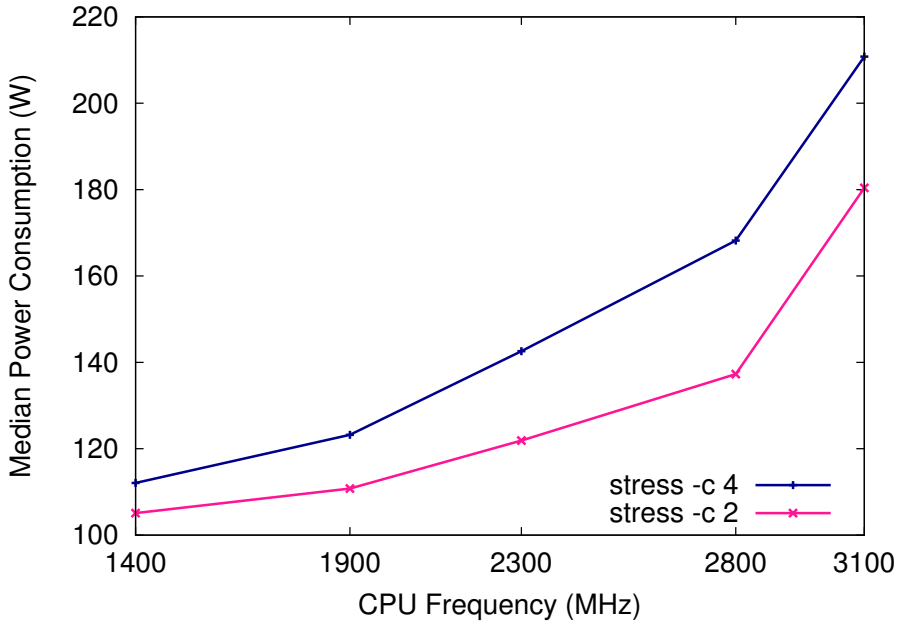

Figure 2.4: Median power consumption during stress on 2 and 4 cores for different CPU frequencies.

setup the median power. Figure 2.5 shows the results.
We notice that the power consumption increases linearly up to two cores being stressed. After that, the power curve gets almost flat. This is due to the fact that there are only two physical cores available on the machine. So even when there are 4 hardware threads, and it seems that there are 4 executions in parallel, the actual power consumed is almost the same as for 2 hardware threads.

2.4 Impact of Temperature

The temperature can have an impact on power consumption and lead to results that may be difficult to interpret. We run a simple utility to stress 5 from 6 cores of the CPU of a AMD Phenom II X6 1090T processor. When measuring the power consumption, we observe severe peaks, as shown in Figure 2.6. This is unexpected, since the workload on the CPU is constant. We would have expected a constant power curve.
Further investigations show that this behavior is related to the temperature of the machine. Using the Linux *sensors* utility, we observe the temperature.
The power consumption reported by the physical powermeter corresponds in the first (increasing) part to the temperature that goes up to 70 degrees. Hardware

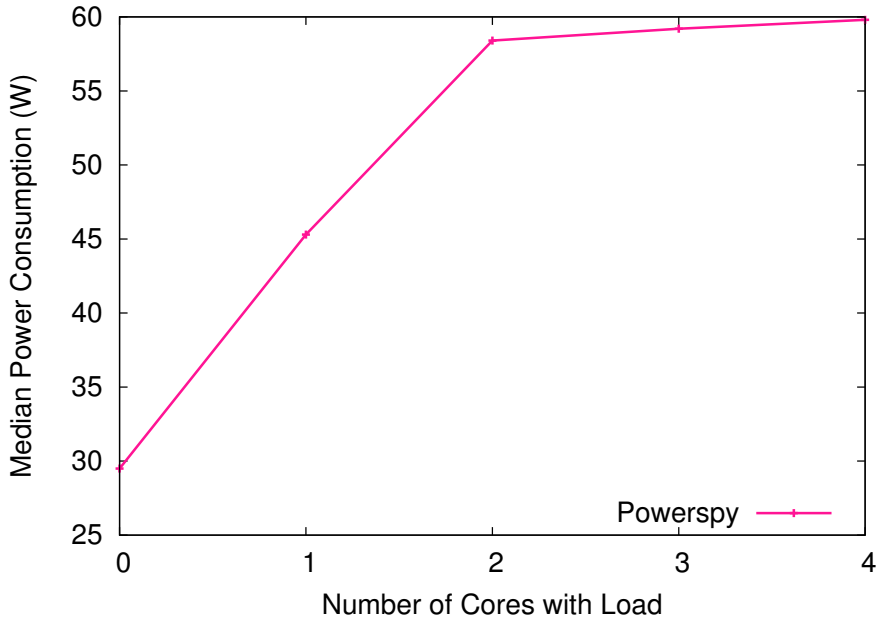

Figure 2.5: Median power consumption for load on an increasing number of cores.

settings prevent the temperature to get higher than 70 degrees and rather decrease the frequency of the CPU. Using the Linux *powertop* utility, we observe that once the maximum allowed temperature is reached, part of each core's load is scaled to a lower frequency. Figure 2.7 shows the temperature during loads of 30 seconds on one to six cores. We can observe that even for this load with a very short execution time, the temperature reaches the maximum when six cores are stressed.

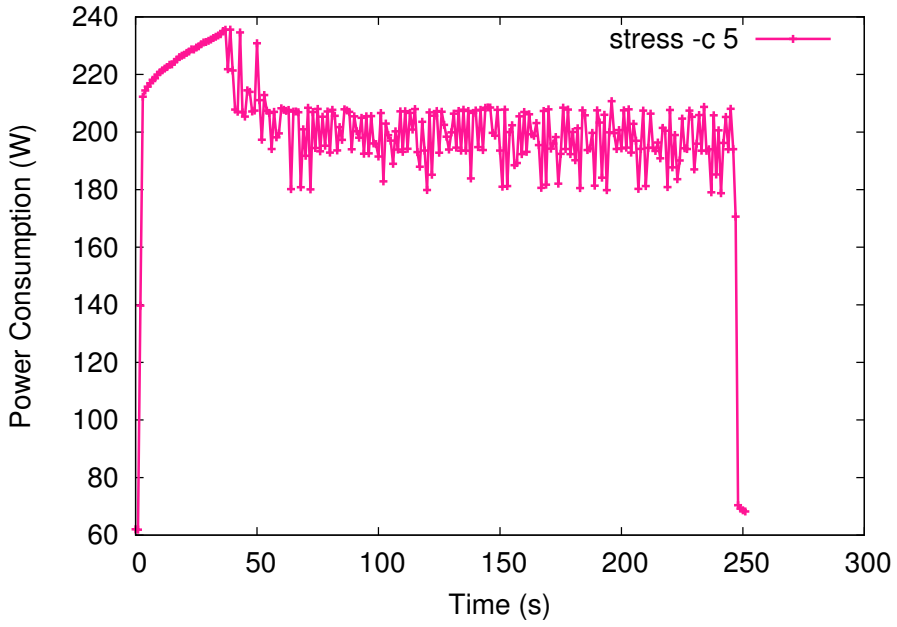

Figure 2.6: Values reported by the physical powermeter for a load on 5 cores with standard settings.

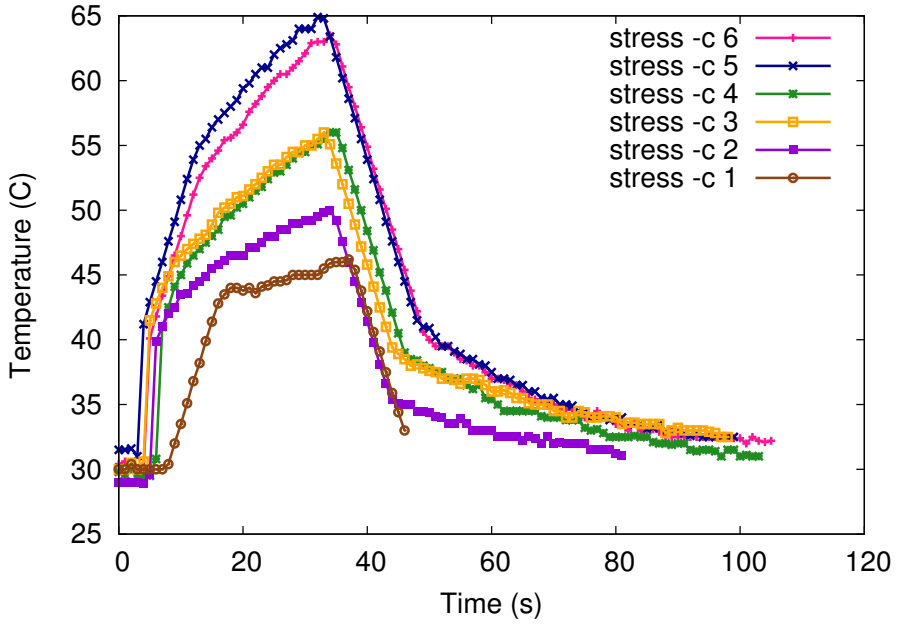

Figure 2.7: Evolution of the temperature during 30 seconds load on a different number of cores.

2.5 The ParaDIME Project

The work presented in this thesis was done in the context of the European Project *Parallel Distributed Infrastructure for Minimization of Energy (ParaDIME)*.[6] Led by the Barcelona Supercomputing Center (BSC), the ParaDIME project consisted of five partners: IMEC (Belgium), Technische Universität Dresden (Germany), Université de Neuchâtel (Switzerland), CLOUD & HEAT Technologies GmbH (Germany) and BSC (Spain). The goal of the ParaDIME project was to come up with a new hardware-software stack, based on emerging device technologies (as heterogeneous processor architectures), new programming models, as well as new runtime and scheduling systems for data centers.

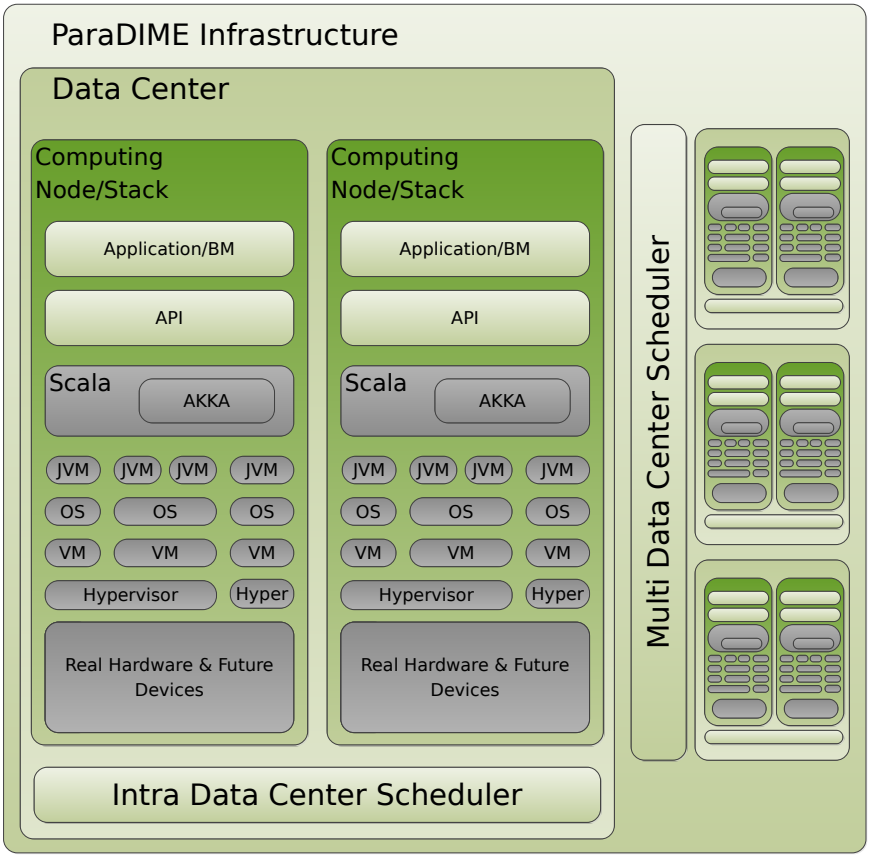

Figure 2.8: The ParaDIME infrastructure.

Figure 2.8 represents the ParaDIME infrastructure. The ParaDIME project combines several methodologies to reduce the energy consumption at different levels. The efforts of the University of Neuchâtel were focused on the benchmark and application

[6] http://www.paradime-project.eu

selection and the programmer's perspective on energy efficiency. In particular, this included: (1) providing a programmable energy-efficient API for message passing, (2) providing support for controlling the voltage of the underlying CPU, and (3) providing an insight on how much energy an application consumes. The contributions of this thesis are relevant in particular for providing an insight on how much energy an application consumes. Regarding the stack in Figure 2.8 my work is placed in the areas of *Intra Data Center Scheduler*, *Real HW*, *Hypervisor*, *VM* and *OS*. The insights I gained during the work on BitWatts [7], EPAVE [5] and PowerIndex [6], which are presented in this thesis, contribute to the goals of the ParaDIME project.

2.6 Related Work

This section contains the state-of-the-art in research related to the contributions of this thesis. In the first part, I will describe existing power models for the CPU and virtual machines. Then, I will look into the related work for data centers considering heterogeneity and power attribution. In the last part I will discuss existing battery models.

2.6.1 Power Models

Models for power estimation have been mainly studied at the level of processors, and less extensively in the context of virtualization. We give an overview of previous research on both aspects in the rest of this subsection.

CPU Power Models

As current platforms do not provide fine-grained power measurement capabilities, McCullough *et al.* [10] argue that power models are the first step to enabling dynamic power management for power proportionality on all levels of a system. Currently, the approach closest to hardware-based monitoring is the *running average power limit* (RAPL) feature available for the Intel Sandy Bridge and Ivy Bridge CPUs [11], which allows for monitoring the power consumption of the whole CPU package.
As this feature is not available on other CPUs, power models typically rely on a number of performance counters. For example Li and John [12] use 5 counters, including the *instructions per cycle* (IPC) counter, and rely on a regression model for estimation. Similar work has been performed by Contreras et al. [13] who additionally consider different CPU frequencies, but not multi-core architectures. Bircher

and John [14] managed to reduce the error of power models to 9 % using performance counters for component-level power estimation. Other work starts with all available counters and then try to reduce their number [15] by analyzing the correlation between counters of different architectures and power dissipation. Usually the accuracy of the models is validated by comparing estimates with the measures of a power meter when running benchmarks in isolation [16].
Power modeling often considers learning techniques—for example based on sampling [17]—that assume the proportionality of system events to power consumption. Measurements of a hardware power meter are gathered and subsequently used, together with a set of normalized estimated values, in various regression models, which are so far mostly linear [10].
Kansal *et al.* [18] and Versick *et al.* [19] notably point out that linear power models depending on the CPU load are not sufficient anymore and that more parameters have to be considered. McCullough *et al.* [10] show that, especially in multi-core systems, linear models lead to a much higher mean relative error of 10-14 % for CPU power consumption and cannot easily be improved by applying more complex techniques. Linear models rely on the independence of the covered features, which is not realistic in current systems. Polynomial/exponential regression can cover these dependencies and, as shown in [14], a quadratic solution better fits the power modeling of multi-core systems. The described systems must however isolate processor features, such as *HyperThreading* and *TurboBoost*, to avoid hidden states. HAPPY [20] introduces a HyperThread-aware power model that differentiates between the cases where either single or both hardware threads of a core are in use.
Shen *et al.* [21] propose power containers to manage energy and power usage on multi-core servers and cloud computing platforms. The authors evaluate power containers with multiple applications, but each of them is considered separately. Power containers are also tied to physical hosts and not evaluated in a virtual environment.

VM Power Models

In data centers, the efficiency of VM consolidation, power dependent cost modeling, and power provisioning are highly dependent on accurate power models [22]. Such models are particularly needed because it is not possible to attach a power meter to a virtual machine [23]. In general, VMs can be monitored as black-box systems for coarse-grained scheduling decisions. If we want to be able to do fine-grained scheduling decisions—*e.g.*, with heterogeneous hardware—we need to be able to consider

finer-grained estimation at sub-system level and might even need to step inside the VM.

So far, fine-grained power estimation of VMs required profiling each application separately. One example is WattApp [24], which relies on application throughput instead of performance counters as a basis for the power model. The developers of pMapper [22] argue that application power estimation is not feasible and instead perform resource mapping using a centralized step-wise decision algorithm.

To generalize power estimation, some systems like Joulemeter [18] assume that each VM only hosts a single application and thus treat VMs as black boxes. In a multi-VM system, they try to compute the resource usage of each VM in isolation and feed the resulting values in a power model.

Bertran *et al.* [17] propose an approach closer to our approach (BitWatts described in Chapter 5). They use a sampling phase to gather data related to *performance-monitoring counters* (PMCs) and compute energy models from these samples. With the gathered energy models, it is possible to predict the power consumption of a process, and therefore apply it to estimate the power consumption of the entire VM.

Another example is given by Bohra *et al.* in [25], where the authors propose a tool named VMeter that estimates the consumption of all active VMs on a system. A linear model is used to compute the VMs' power consumption with the help of available statistics (processor utilization and I/O accesses) from each physical node. The total power consumption is subsequently computed by summing the VMs' consumption with the power consumed by the infrastructure.

Janacek *et al.* [26] use a linear power model to compute the server consumption with *postmortem* analysis. The computed power consumption is then mapped to VMs depending on their load. This technique is not effective when runtime information is required.

In Stoess *et al.* [27] the authors argue that, in virtualized environments, energy monitoring has to be integrated within the VM as well as the hypervisor. They assume that each device driver is able to expose the power consumption of the corresponding device as well as an energy-aware guest operating system and is limited to integer applications.

2.6.2 Data Centers

Heterogeneity and Placement

Existing research often assumes that data centers consist of homogeneous hardware. However, this is not realistic in real-world data centers, as shown in [4]. We often find different generations of the same hardware in a data center, or even different types of hardware combined. This tendency led to the concept of *warehouse scale computers (WSCs)* [28] composed of diverse microarchitectures and configurations.
Paragon is a scheduler aware of heterogeneity and interference [29] [30]. It decides for a specific workload on which server configuration it will perform best (heterogeneity). It also considers how much interference it will cause to other applications and how much interference it can tolerate regarding its own shared resources. This approach mainly focusses on performance and preserving the QoS rather than power or energy efficiency.
Another existing approach focusses more on the power aspect: It defines an intelligent workload allocation method that efficiently maps workloads to the best matching platform [31]. The authors show that their approach significantly reduces the overall power consumption of the data center (12-22% power saving over random allocation). In their evaluation, they measure power and performance across each platform and then extrapolate the measured data using a data center allocation simulator.

Idle Power Attribution

The idle power concerns the power consumed by an infrastructure which is powered on but not running any task. Typically, for a server, it consists of the energy consumed while idle, but fully powered on. This consumption depends on the hardware of the server, but it can also depend on the operating system installed on it as it is responsible for the background tasks running continuously on the server (like monitoring tasks). This power is usually not taken into account by VM-based models. For an entire data center, the idle power consumption includes all the power which does not depend on the workload.
Often only the power consumption of IT equipment is considered although air conditioning can consume 33% of the global power needed by a data center [32]. This cost can be reduced by free cooling techniques exploiting outside air [33]. The power consumption of such cooling techniques is tightly correlated to the weather, and thus vary over time even if the workload does not vary. Therefore, their power

consumption, which is considered to belong to the static idle part, can vary over time.
Most of the studies do not use the same definition for the energy costs of the computing infrastructure: for instance, the network used to link the computing resources is not taken into account most of the time. In the same way, as surveyed in [34] some works take into account only the dynamic consumption of the machines and not their static consumption (corresponding to the consumption when machines are powered on but idle) which can yet represent more than half of the total power consumption. In our context, we will consider all the equipment operated by the cloud provider: the data centers (including cooling infrastructures) and the network links inside and between the data centers.

Power Attribution Models

Benefiting from economies of scale, cloud infrastructures can effectively manage their resources and offer large storage and computing capacities while minimizing the costs for users. However, the rapid expansion of these infrastructures led to an alarming and uncontrolled increase of their power consumption. For instance, in 2010, the services offered by Google were based on 900,000 servers that consumed an average of 260 million Watts [35].
Moving from instrumenting to modeling the energy consumption is a tough but necessary task in order to improve the energy efficiency of distributed infrastructures. It is indeed essential to understand how the energy is consumed to be able to design energy-efficient policies.
Most of the models found in literature split the consumption of an entire server into the consumption of each component of the server [25] or consider that consumption is proportional to the load [36]. Several studies are focused on modeling the energy consumption of particular components: CPU [37] influenced by the frequency, voltage and workload, network card [38] with costs per packet and per byte, and disk [39] driven by the rotational speed and the read and write operations.
However, as shown in [40] the limits of these approaches are to model the energy consumption of entire servers under various workloads. Concerning the experimental approaches found in literature, they mainly consider just one type of machine, or even only one type of application [40]. So, it is necessary to design unified models closer to reality. Concerning the consumption of entire infrastructures, the authors of [41] show that computing resources represent the biggest part in cloud's consumption. An alternative approach [42] shows that, depending on the studied scenario, the energy

costs of the network infrastructure that links the user to the computing resources can be bigger than the energy costs of the servers.
As shown in [40], simple models are not convincing in the general case and especially for multicore architectures – which tend to become widespread. It is therefore necessary to depend on benchmarks for the development and validation of reliable energy cost models for these heterogeneous resources. These benchmarks need to propose several kinds of workloads: computation-intensive, disk-intensive, etc.

2.6.3 Battery Models

Apart modelling the components with power plug, we need to consider mobile devices that might be powered by battery.
The most simplified battery model consists of an ideal voltage source in series with an internal resistance [43]. The drawback of this model is that it does not include the state-of-charge of the battery.
Chen et al. [44] propose a model based on usable capacity, open-circuit voltage, and transient response and implement it in the *Cadence* environment. The model accounts for all dynamic characteristics of the battery, from non-linear open-circuit voltage, current, temperature, cycle number, and storage-time dependent capacity to transient response. In the validation they are using a simplified model: it neglects self-discharge, cycle number, and temperature. The evaluation considers NiMH and polymer Li-ion batteries. Erdinc et al. [45] extend the previous model and establish a simulation model in a *MATLAB/Simulink* environment. The extension of the model consists in adding the effects of temperature and capacity fading. The drawback of these two approaches are that the parameter extraction requires experiments on physical batteries for each type of battery.
Kuhn et al. [46] propose a model based on open circuit voltage in series with a resistance and parallel RC circuit with Warburg impedance. The identification of the parameters for this approach is based on impedance spectroscopy [47], and thus rather complicated.
Badam et al. [48] implement a system that can integrate batteries of different chemistries. The approach is based on open circuit potential, internal resistance, concentration resistance, and plate capacitance (simplified Thevenin model). It consists of components across three layers: batteries and their chemistries, the battery management circuit, and the operating system. The software-defined battery (SDB) hardware supports discharging and charging across multiple heterogeneous batteries. Authors provide a hardware prototype and a multi-battery emulator to evaluate the solution.

The drawback of this solution is that the parameter extraction is complex. The four parameters required are learned during experimentation on physical batteries and, as a consequence, it might be hard to apply this model to different types of batteries. Gao et al. [49] propose a model that accounts for non-linear equilibrium potentials, rate and temperature dependencies, thermal effects, and response to transient power demand. The model is based on publicly available data (e.g., from datasheets). This approach claims to be an intermediate approach with sufficient accuracy, but avoiding detailed calculations of internal electrochemical processes. The model was tested in the *Virtual Test Bed* environment.
Our model which is described in detail in Chapter 6 is based on previous work of Tremblay et al. [50] [51]. Their model is based on the Shepherd model [52], with some modifications due to implementation factors. The model from Tremblay et al. is included in the *SimPowerSystems* simulation. The parameter extraction is very simple and all required parameters can be extracted or easily computed from the manufacturer's datasheet. The battery model is based on a simple controlled voltage source in series with a constant resistance. The open voltage source is calculated based on the state of charge. The discharge behavior for a Li-Ion battery is modelled by formula:

$$V_{batt} = E_0 - R \cdot i - K\frac{Q}{Q - it} \cdot (it + i*) + A_{exp}(-B \cdot i)$$

where V_{batt} is the battery voltage (V), E_0 is the battery constant voltage (V), K is a polarisation constant (V/(Ah)) or polarisation resistance (Ω), Q is the battery capacity (Ah), it is the actual battery charge (Ah), A is the exponential zone amplitude (V), B is the exponential zone time constant inverse $(Ah)^{-1}$, R is the internal resistance (Ω), i is the battery current (A), and i^* is a filtered current (A).

3 Using Power Measurements as a Basis for Workload Placement in Heterogeneous Multi-Cloud Environments

In this chapter we study the performance and energy efficiency of a set of heterogeneous architectures for multiple applications. This chapter covers the first part of the contributions for the data center layer of the stack presented in Chapter 1. More sophisticated approaches for data centers will be presented in Chapter 4.

3.1 Introduction

Due to the environmental concerns of different energy resources and the massive power consumption of large data centers, energy efficiency becomes more and more important. Cloud computing as a whole consumes more energy than countries like Germany or India [1]. Besides the environmental reason, reducing the energy consumption within data centers reduces their total cost.
A first step to reduce energy consumption is making hardware more energy-efficient. One of the hardware components responsible for the highest power consumption is the CPU. The Thermal Design Power (TDP) specifies the maximum amount of heat generated by the CPU that must be dissipated by the cooling system. For example, an Intel i7 Bloomfield processor with 4 cores from 2008 has a TDP of 130W.[1] In contrast, an Intel i7 Haswell-DT processor with 4 cores from 2013 has a TDP as low as 35W.[2] An approach to improve energy efficiency is to reduce energy and power consumption by energy-aware workload-scheduling. For example, Mashayekhy et al. [53] propose a scheduler where MapReduce jobs are scheduled to different machines under

[1] http://ark.intel.com/products/37147/Intel-Core-i7-920-Processor-8M-Cache-2_66-GHz-4_80-GTs-Intel-QPI, accessed on 17.07.2016
[2] http://ark.intel.com/products/75121, accessed on 17.07.2016

constraints of energy consumption. The authors developed a greedy algorithm that improves energy efficiency while still satisfying the negotiated service level agreements (SLA). With their approach, the energy consumption can be reduced by 40 %, but they only consider a single data center and assume homogeneous hardware.
The concept of *sky computing* was introduced by Keahey et al. [54]. The authors show that with virtualization and overlays it is possible to build a multi-cloud environment in a trusted environment. However, this brings also problems of heterogeneity (not only on the hardware level) and application scheduling becomes more challenging. One example that tackles multi-cloud scheduling is shown by Tordsson et al. [55], who optimize the placement of virtual machines among different cloud providers to achieve higher performance, lower costs and better load balancing. A *cloud broker* containing a scheduler implements the decision logic based on user-specified criteria, but does not in particular consider heterogeneous hardware or energy efficiency. In the Reservoir project [56] a key role is a scheduler that assigns a particular workload to the *best* fitting cloud. The placement is based on a *load balancing policy* or a *power preservation policy*. With the power preservation policy the virtual machines are aggregated on a minimal number of physical machines and the other machines are switched off. These two policies show that there is a tradeoff between performance and power consumption. While the reduced number of machines will reduce the overall power consumption, the challenge is to not over-commit the system to avoid SLA violations during peak phases.
Besides tackling challenges towards performance and power consumption, most of the previously described research assumes that data centers consist of homogeneous hardware. However, this assumption is not realistic in real-world data centers, as shown in [4].
In this chapter, we show that it is important that the scheduler is aware of the existing hardware as well as of the type of workload. As a basis, the scheduler needs information on the current power consumption as well as a notion of current performance. We show that performance per Watt is sufficient for categorizing different resources. We categorize workload types that are either CPU-intensive or disk-intensive. Then we run them on different setups and show that for being energy-efficient, workloads have to run on the right type of hardware. This experimental study is the first step towards energy-efficient scheduling decisions in a heterogeneous multi-cloud environment.
This chapter is organized as follows. In Section 3.2 we explain the experimental setup with the hardware and software specification. We further go into details on the metrics used. Section 3.3 contains an extensive discussion of the gathered results and finally we conclude the chapter.

3.2 Experimental Setup

For our experiments we selected a number of hardware setups covering recent and older architectures as well as a number of representative workloads, which we describe in the following sections.

3.2.1 Hardware

Entries numbering from 1 to 8 in Table 2.1 in Chapter 2 give an overview of the architecture characteristics of a variety of systems that comprise either AMD CPUs or Intel CPUs. These machines were used in the evaluation of this chapter.
Besides the TDP, the idle power is an important characteristic to determine energy efficiency. The idle power is the power required to run just the operating system on the hardware, hence it is typically constant and has to be considered for every workload. We therefore compare the idle power of the selected systems.
For all of our measurements, we use a highly-sensitive physical power meter (PowerSpy).[3] The driver for Linux is available as open source software.[4] In Figure 3.1 we see that the AMD with 387 W and the Xeon with 208 W consume already much more idle power than the other systems. The reason is that these are server architectures that do not comprise the most recent hardware parts. In comparison, the mobile devices (i7 and VIA) have a very low idle value (11W and 19W respectively) as their main target is to enhance battery life.

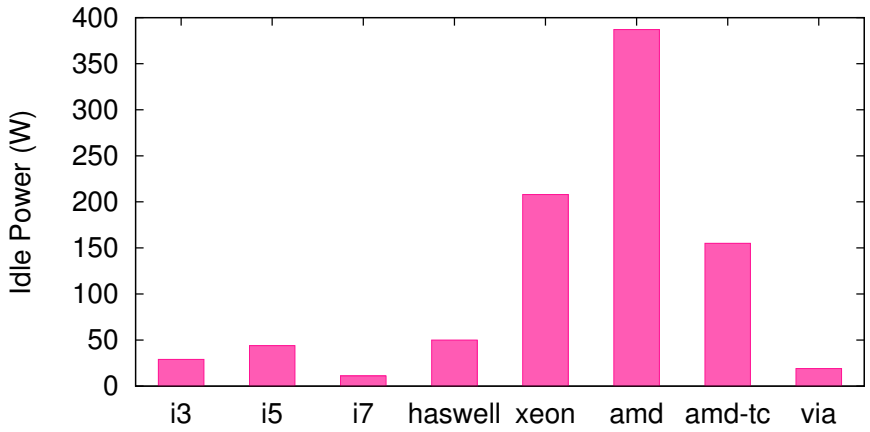

Figure 3.1: Idle power consumption (only OS, no workload) of the different systems.

[3] http://www.alciom.com/fr/produits/powerspy2.html, accessed on 17.07.2016
[4] https://github.com/patrickmarlier/powerspy.py

3.2.2 Workloads

We selected workloads that can be categorized as CPU-intensive or disk-intensive to understand the behavior of the different systems under different loads.
For the CPU-intensive tasks we decided to go for a variety of microbenchmarks, and one real-world application as power consumption cannot be linearly linked to CPU-utilization. For the disk-intensive workload, we investigate performance and power consumption during different, but controlled operations on the hard disk.
Stress. The *stress*[5] utility is a workload generator, which we use as microbenchmark to stress CPU and memory. The CPU stress loops on the *sqrt()* operation for a given time period (30 seconds). We allocate 1 to N workers, with N being the number of threads (or cores without HyperThreading) available in the system. To stress the memory, we create a stress worker that spins on *malloc()* and *free()* operations. It allocates 1 to M GB, where M is the number of GB of RAM available minus one. By keeping 1 GB for the system, we avoid swapping of memory during the experiments, which would influence the system behavior and hence the power measurements.
Factorial. The stress workload runs for a given time period. Thus, depending on the capabilities of the different architectures it will perform a different amount of work. In contrast, each machine will perform the same number of operations when performing a complex computation. In our second benchmark we calculate the factorial of a large number (299,999).
SPECjbb2013. As a real-world benchmark, we use SPECjbb2013 [57]. This benchmark is implemented in Java and represents a typical software for a supermarket company, including distributed warehouses, online purchases and high level management operations such as data mining. During the execution, it covers different levels of CPU utilization. To ensure the benchmark runs smoothly, the JVM gets 3 GB memory per run. We exclude the VIA architecture from these experiments as it does not fulfill the minimum requirements of the benchmark in terms of memory.
Bonnie++. We use *Bonnie++*[6] to stress the disk. Bonnie++ has different phases including sequential output, sequential input and random seeks. In the phase of sequential output it writes single characters and entire blocks, and it modifies blocks. For the sequential input it reads character by character and entire blocks. In the end it comprises a phase of random seeks where the benchmark measures the physical movements of the head of the hard disk.

[5]http://manpages.ubuntu.com/manpages/trusty/man1/stress.1.html, accessed on 17.07.2016
[6]http://sourceforge.net/projects/bonnie, accessed on 17.07.2016

3.2.3 Metrics

Every second, the PowerSpy power meter reports its measurements in Watt. If the power consumption of a workload is steady, we can take the median power consumption P and multiply it by the execution time T of the workload, which results in the energy consumption E reported in Joule:

$$E = P * T$$

We can use this metric for the stress experiment, as its resource usage is steady. Whereas the median power consumption makes sense for the constant load in the stress experiment, it cannot be used for the other experiments. In the evaluation using Bonnie++, SPECjbb and the factorial *performance per Watt* as a metric is more informative. Performance per Watt is a performance metric (usually throughput) divided by the power consumption:

$$\text{Perf}/W = \frac{\text{Throughput}}{P}$$

For the factorial experiments, the throughput is defined as the number of iterations divided by the execution time. The throughput for the disk experiments with Bonnie++ is the write or read rate per second. For the SPECjbb benchmark, we use the maximum number of jOPS, which is a throughput metric provided by the SPECjbb benchmark [57]. It represents the overall maximum throughput capacity of the system in terms of response-throughput.
The power consumption includes also the idle power as we want to observe the total cost in terms of power when running the workload on a specific hardware.

3.3 Results

In this section we will show that two types of workloads, CPU-intensive or disk-intensive, have different power characteristics and can hence best take advantage of different hardware configurations.

3.3.1 CPU and Memory Power Consumption

To classify the different CPUs, we show in Figure 3.2 the median power consumption if we apply the stress command on all cores for 30 seconds. In the case of Hyper-Threading, we stress up to the number of available threads. We observe similar power consumption curves for the i3, the i5, the i7 and the Haswell systems. The power

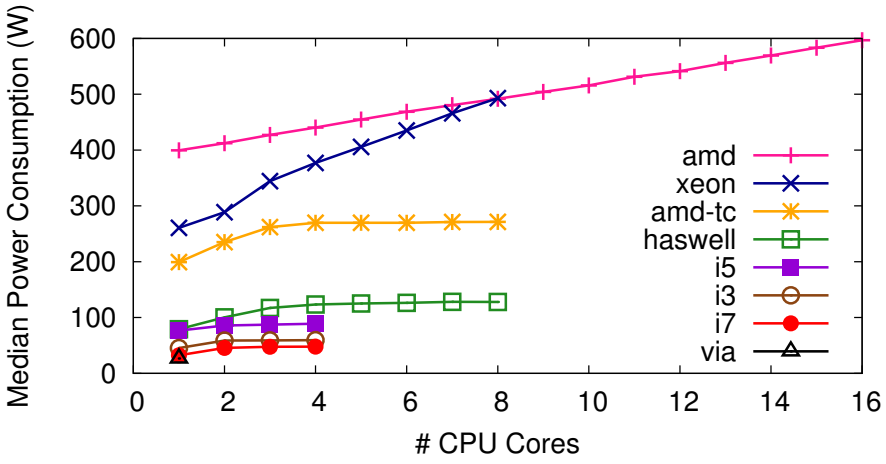

Figure 3.2: Median power consumption for CPU stress on all available cores.

consumption increases from one to two cores (or from one to four cores in the case of Haswell), and is then constant. Hence, the power consumption only increases up to the number of physical cores. The power consumption for the AMD is almost linear to the number of cores. We assume that this is caused by the rather simple architecture of the CPU with fewer hardware features (i.e., no HyperThreading, no turbo feature, etc.).

To provide more details about stressing single cores, Figure 3.3 depicts the power consumption for the i3. We stress one to four cores and we see that for stress on one core, the power is significantly lower than for the multiple core stress. If we stress two cores, we see that the scheduler tries to allocate two separate physical cores for load balancing. Therefore, the power consumption is doubled. However, this behavior could be influenced by pinning the stress processes to specific cores or HyperThreads. The difference between two and four cores is very small because of HyperThreading.

To compare the performance per Watt of the CPU workload, we measure the power of the computation of the factorial for a six-digit number (299,999). The number of iterations is constant, however, the execution time varies depending on the capabilities of the given system. Figure 3.4 depicts the execution time and the median power for the different systems. The VIA has a very long execution time for the factorial computation, but a very low median power consumption. When comparing the Haswell (4th generation i7) and the i7 (2nd generation i7), we notice that unsurprisingly the older processor is slower. When considering the median power, the older i7 consumes around 30 W whereas the Haswell machine consumes almost

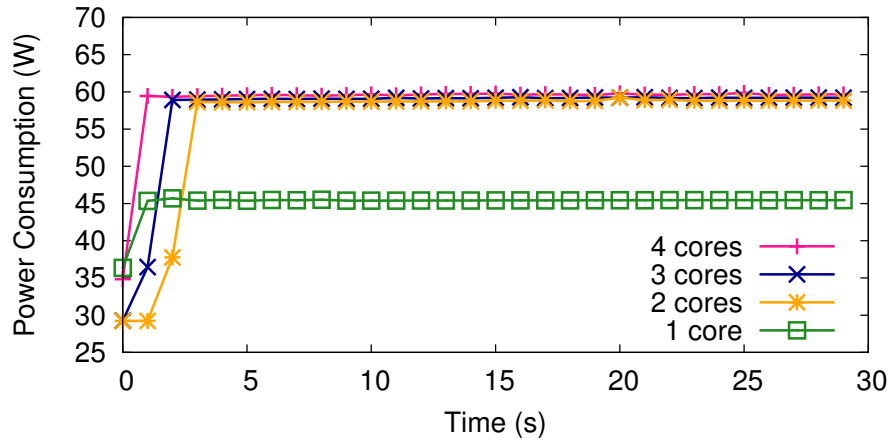

Figure 3.3: Power values for the CPU stress on 1-4 cores on the Intel i3 machine.

80 W. The results of the Xeon and the AMD show a high impact of the high idle power in their final results. Hence, considering power consumption, it is expensive to run CPU-intensive applications on these machines. Furthermore, the AMD has a lower CPU frequency (2.2 GHz) and thus a longer execution time for the factorial computation.

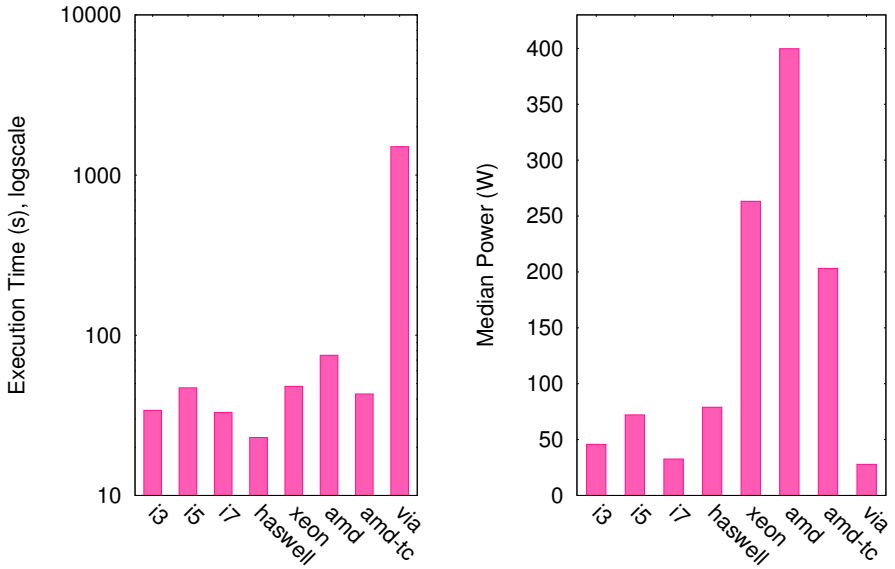

Figure 3.4: Execution time and median power of a factorial computation on different machines.

In order to achieve a good performance per Watt, machines need to provide a good

tradeoff between execution time and power consumption. Figure 3.5 shows the performance per Watt for the factorial computation. Due to their long execution times, the VIA and the AMD have a very poor throughput per Watt, which is less than 20 iterations/s for a Watt. The i3, the i7 and the Haswell provide a good tradeoff between execution time and power and thus have a good performance per Watt. The i7 of 2nd generation is more energy-efficient than the Haswell (4th generation i7), even though the execution time of the i7 is higher. We can thus conclude that the power consumption can be more important than the processor performance for energy efficiency.

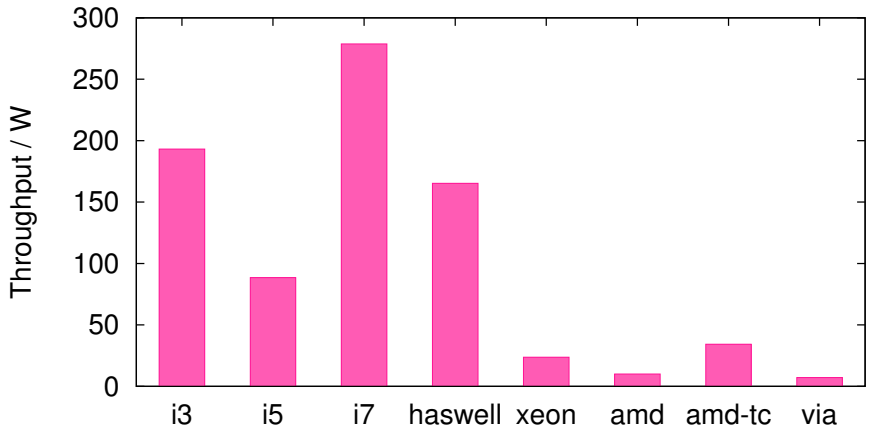

Figure 3.5: Throughput per Watt for the factorial computation (based on median power consumption).

Most workloads do not only stress the CPU, but also allocate memory. Hence, in Figure 3.6 we present the median power consumption for a memory-intensive workload.

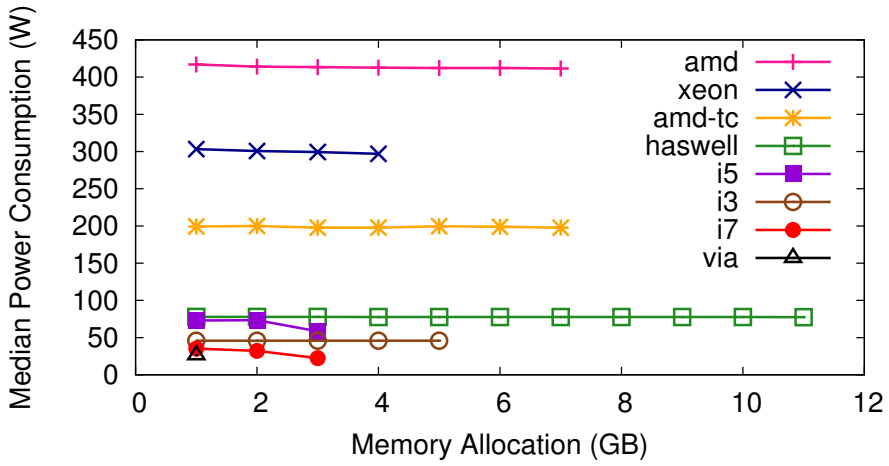

Figure 3.6: Median power consumption for memory stress from 1 GB to the number of GB of RAM available minus one.

We can observe that the power consumption is rather constant when a process is writing to memory. For the i5, we see that the power consumption decreases for a memory allocation of 3 GB. This machine started swapping and thus was less efficient.

To validate these results with real-world applications, we run the SPECjbb2013 benchmark [57]. Figure 3.7 shows the power consumption of the SPECjbb2013 workload on the i3 during a benchmark run. In this run, it is possible to identify the different phases of the workload just by plotting their power consumption. SPECjbb2013 consists of the following phases: (1) search HBIR (High Bound Injection Rate), (2) RT curve building (Response Throughput), (3) validation (run checks), (4) profiling (statistical data), and (5) reporting.

The *search HBIR* phase approximates the maximum injection rate the system can handle. The most important phase for our evaluation is the *RT curve building* phase. Starting from 0% of HBIR, it increases the injection rate step by step until the maximum capacity is reached. This phase provides the data that is used to determine the maximum jOPS, which we use as throughput metric for the evaluation. The RT curve building phase can be observed in Figure 3.7 between second 600 and second 1300.

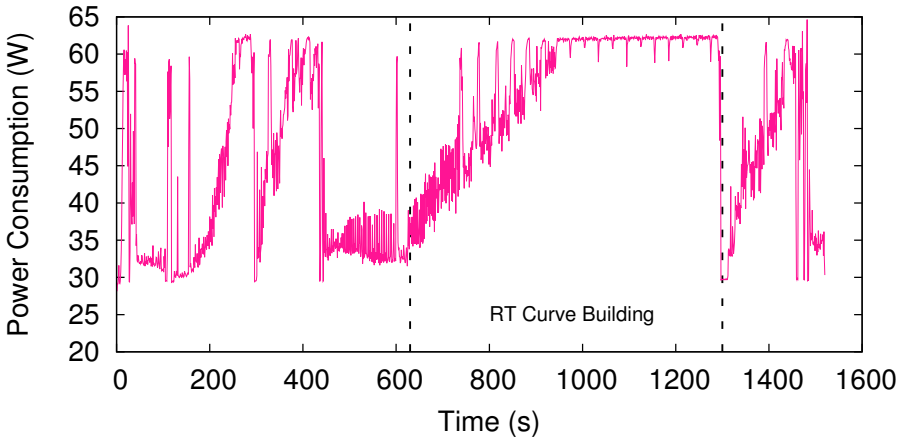

Figure 3.7: Power consumption of the SPECjbb workload on the i3 machine.

Based on the maximum jOPS and the maximal power values reached during the workload, we evaluate the performance per Watt. Figure 3.8 shows the maximum jOPS reached on the different architectures. The Haswell processor achieves the highest number of jOPS (almost 13,000). Other architectures like the i3 achieve less than 4,000 jOPS. Since the different architectures reach different jOPS, we cannot compare the execution time or the energy. With these two metrics, the machines reaching a higher throughput would be penalized.

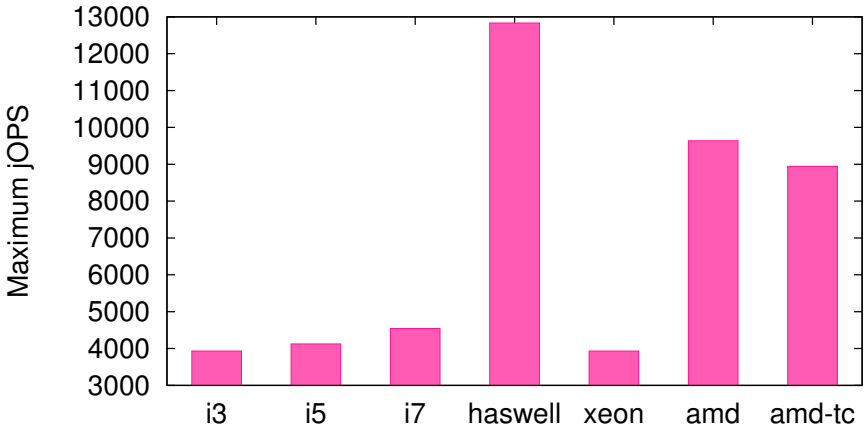

Figure 3.8: Maximum jOPS for the SPECjbb workload.

Figure 3.9 depicts the maximum power value reached by each architecture during the execution of the SPECjbb2013 workload. The Xeon, even though not reaching a high number of jOPS, has a very high maximum power.

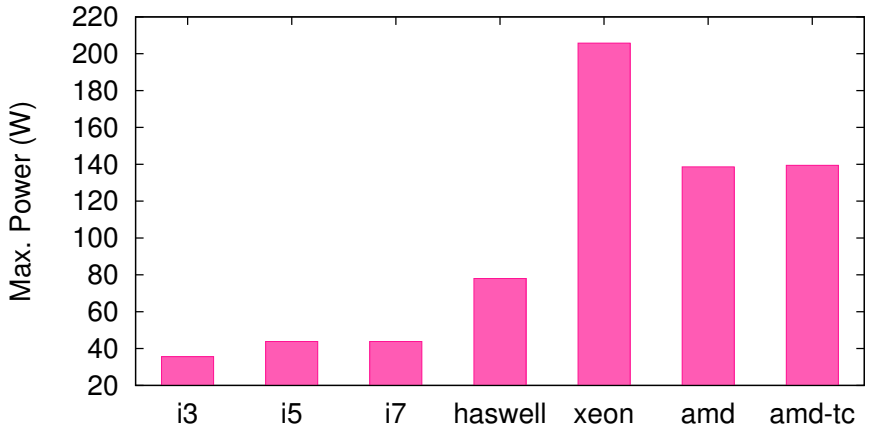

Figure 3.9: Maximum power value in main workload phase for SPECjbb workload.

Therefore, as shown in Figure 3.10, it has a very low performance per Watt. The Haswell machine with very high jOPS and a relatively low power consumption has the highest performance per Watt for the real-world workload.

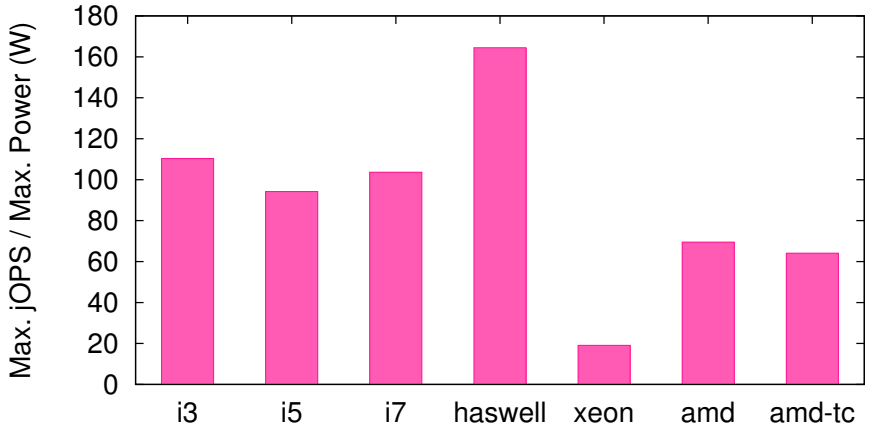

Figure 3.10: Performance per Watt for SPECjbb workload.

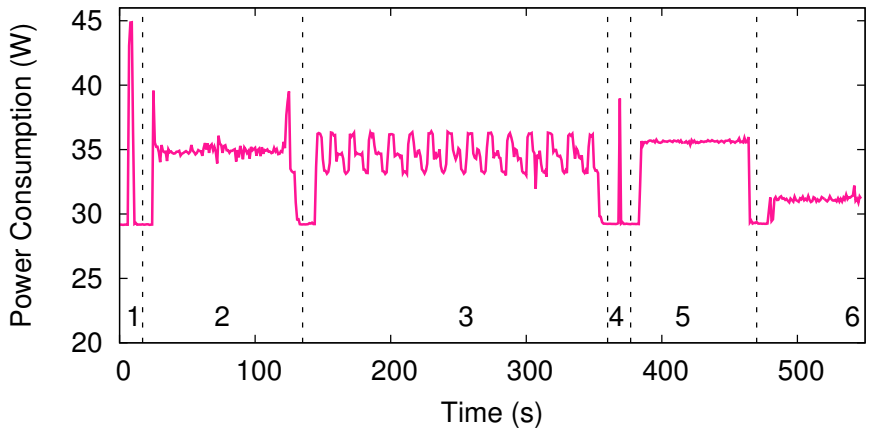

Figure 3.11: Bonnie++ workload for 1) sequential write by character, 2) sequential write by block, 3) modifying of blocks, 4) sequential read by character, 5) sequential read by block and 6) random seeks on the i3.

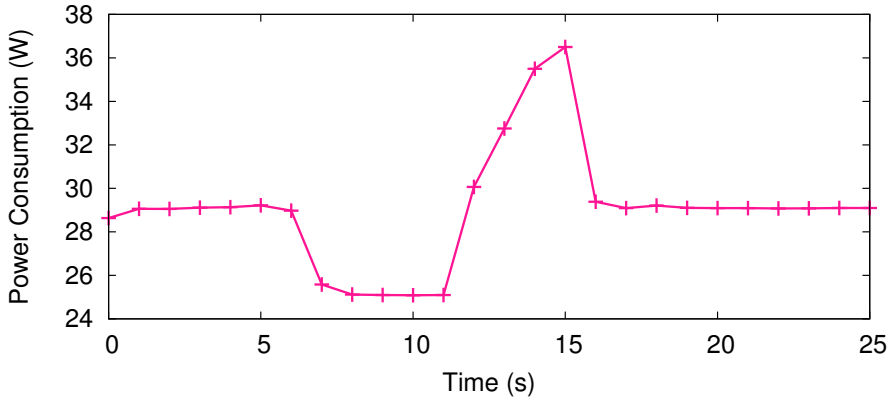

Figure 3.12: Disk on i3 put to standby and turning back to active state.

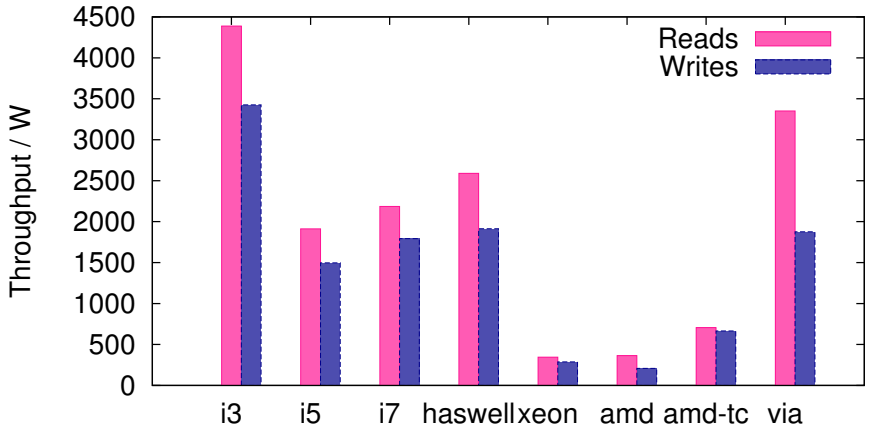

Figure 3.13: Write and read rate respectively divided by median power during execution of Bonnie++ during phases of block writing and block reading.

3.3.2 Disk Power Consumption

Some workloads, like for example webservers and databases, do not only utilize CPU and memory but write also to disk. We need to include such workloads in the power evaluation, too.

Figure 3.11 shows the power consumption for the Bonnie++ workload. This disk workload consists of the following phases: (1) sequential writing by character, (2) sequential writing by block, (3) modifying blocks, (4) sequential reading by character, (5) sequential reading by block, and (6) random seeks. The phases are separated by sleeps of 15 seconds.

When writing sequentially to the disk, we notice peaks in the beginning and in the end of the write operation. To the best of our knowledge, the initial peak is caused by the disk changing from the standby state to the active state. We verify our hypothesis by using the *hdparm* utility to put the hard disk in standby mode (see Figure 3.12). The figure shows that when putting the disk in standby, we save 4 W. To change the state from standby to active, we require for a short time 8 W.

To compare the different architectures, we study the performance per Watt for block reads and writes, which we depict in Figure 3.13. The executions of writes and reads by character are too short and the results of the block modifications were not comparable as they are not stable enough.

We observe that the VIA netbook achieves a quite good performance per Watt, in particular for reads from the disk. This can partially be explained by the small disk

size of 120 GB and a lower RPM. As seen in the previous experiments, systems with a high power have a lower performance per Watt.

3.3.3 Discussion

We notice that the same architecture can be more efficient for disk workloads and less efficient for CPU- or memory-intensive workload. In terms of throughput per Watt for the factorial computation, the i7 is the most energy-efficient architecture (throughput per Watt 279), 44.5% better than the i3 with 193. For the disk workload, the i3 and the i7 have a throughput per Watt of 4388 and 2185 respectively, thus in this case the i3 is twice better than the i7. An energy-efficient scheduler needs thus to make the right decisions based on the characteristics of the workloads.

AMD	**i3**	**Haswell**	**Total(J)**
0	5xdisk	5xCPU	14370
0	5xCPU	5xdisk	16110
5xdisk, 5xCPU	0	0	210130

Table 3.1: Different workload placements and total energy costs in a fictive data center.

To give a feeling about scheduling impact, we consider the scheduler proposed by the Reservoir project [56] and its power preservation policy as explained in Section 3.2.2. Virtual machines are aggregated on physical hosts, and unused machines are turned off. We imagine a fictive data center with machines from our configuration: 10 times AMD, 5 times i3, 5 times Haswell. Based on how the scheduler assigns the workloads, different power scenarios are possible. We consider a very simple scenario where we place one virtual machine on each physical host and each virtual machine contains a single workload. There are ten workloads to be scheduled, half of them are writes of 30 seconds to the disk and the other half are CPU/memory-intensive factorial computations of a large number. The factorial computation consumes 1,555 J on the i3, 1,810 J on Haswell and 30,018 J on AMD. Writing for 30 seconds to the disk consumes 1,064 J for the i3, 1,667 J on the Haswell and 12,008 J on the AMD.
Table 3.1 shows possible placements by a scheduler and the costs in terms of energy. We see that the total energy drastically changes based on the scheduler decision: in the best case, the total cost is 14,370 J, whereas in the worst case it is 210,130 J, which is 14 times more. In a heterogeneous data center it is essential to choose the right hardware for a given workload. In particular, a scheduler needs to focus

on aggregating workloads on energy-efficient machines and possibly turn off servers with high energy consumption.
The cross-cloud broker could also include the facility to aggregate power estimation values from the different data centers and different physical machines. These values would be stored and processed in the broker and used as a basis for scheduling decisions.

3.4 Summary

In this chapter we studied the energy efficiency of different workloads on heterogeneous hardware. This information is relevant for schedulers in both heterogeneous data centers and multi-cloud environments.
Our results show that the characteristics of the workload and the machine are important for cost reduction in data centers. For example, a machine that has a high throughput per Watt for disk access does not necessarily have energy-efficient results for a CPU-intensive workload.
In an example of a fictive data center with a very simple scheduler, we showed that we consume 14 times less energy in the best case than in the worst case. Thus, energy efficiency is clearly an important decision metric for schedulers in a heterogeneous context.
Such a scheduler can take advantage of different workload characteristics and the availability of heterogeneous resources. Workloads can then be classified based on parameters such as runtime, priority and usage of resources (e.g., CPU, memory or disk). The scheduler would then choose for each category the most energy-efficient hardware available for deployment. Even in a federated multi-cloud environment it would be advantageous for each cloud provider to have its data center filled with the most energy-efficient workloads. When workloads are distributed in an energy-efficient manner on the available hardware resources, the total cost of the data center can be reduced.

4 How much does a VM cost? Energy-Proportional Accounting in VM-based Environments

In this chapter we present EPAVE, a model for energy-proportional accounting in VM-based environments, and PowerIndex, a profiling and energy estimation model. This chapter is the second part of contributions for the data center layer of the stack presented in Chapter 1. The results presented in Chapter 3 are the basis for the approaches presented in this chapter.

4.1 Introduction

The trend of computing in the cloud grows, which consequently requires bigger data centers, more processing power and hence more CPUs. Whereas the hardware gets more and more energy-efficient the overall energy consumption of data centers increases. Actually, today's cloud computing requires more electricity in the form of energy than entire countries such as India or Germany [1].

It is hence not surprising that energy represents one of the main cost factors of a data center. The major energy consumers are the air conditioning, the network infrastructure (routers, switches) and the servers [34]. However, these costs are rarely reflected in the attribution of energy consumption to a single consumer (e.g., a virtual machine).

As users share the same resources on a single node, most of the existing models concentrate on attributing the power consumption of this shared node to a single consumer. For instance, how much of the CPU power consumption can be related to a VM [25], [18], [7]?

Our vision is to consider energy accounting on the data center level to enable pricing models where every user will pay for the actual usage of resources. The first challenge is to provide a fair attribution model that is predictable to provide incentives for

energy-efficient computing in the cloud. The second challenge is to consider the *mobility* of VMs. A VM can be easily spawned on a different node, which might have different hardware specifications. These different hardware specifications might lead to different energy behavior.
In this chapter we tackle these two challenges, by

1. showcasing EPAVE (*E*nergy-proportional *P*rofiling and *A*ccounting in *V*irtualized *E*nvironments) for realizing accounting of real energy costs of the data center to each client considering the major consumers and the entire facility costs and

2. extending EPAVE with PowerIndex that allows us to predict the energy consumption of the same VM on different hardware.

More specifically, we target energy proportional accounting for each virtual machine (VM) in a heterogeneous environment.

Context. Currently the relation between IT infrastructure and facility energy costs are modeled with the Power Utilization Efficiency (PUE) metric. This metric is used to help operators on decisions regarding new hardware infrastructure. For instance, Google measures the PUE per site each three months[1] for each of its data centers. While the PUE is a useful metric for reflecting the overall efficiency of a data center, its applicability for the day-to-day operation of the data center is limited because it does not grasp the variability of the actual power consumption of the data center. In a data center, the instant power consumption can be divided into static and dynamic parts. The static parts are the base costs of running the data center when being idle; the dynamic costs depend on the current usage. In an ideal case, the overall power consumption would be proportional to the utilization of the hardware (power proportionality). However, having non-negligible static parts, power proportionality is not yet achievable [58]. Nonetheless, we can get closer to power proportionality by accrediting the static power parts to each application, depending on the time and the resources used. Since time plays a major role, we will focus on energy instead of power consumption (an instant measure). Hence, we talk rather about energy proportionality than power proportionality.

Challenges. Dynamic power consumption mainly depends on the resources which are used: computing, storage, networking resources. In the case of virtual environments, the hardware resources may be shared among different users and different

[1] http://www.google.com/about/datacenters/efficiency/internal, accessed on 17.07.2016

virtual machines, if they run on the same host. In this context, a power-aware model needs to estimate the relative utilization per user to attribute the dynamic costs of the physical resources to a particular VM.
The static costs have to be considered at different levels:

- at the data center level: power delivery components, cooling systems, other miscellaneous components such as data center lighting. This part is captured by the PUE.
- at the resource level: idle power consumption of servers and routers.

The main challenge we tackle in this chapter is to divide the static costs among the users in a fair and predictable way, considering the utilization of the resources per VM. We have shown in [5] that a simplistic model is not enough for distributing the costs among a number of VMs as the static costs would be highly dependent on the utilization of the same server.
As for dynamic costs, they can vary significantly from one server architecture to a different one. Performance and energy consumption heterogeneity among the servers is inherent to cloud data centers. Typically, 3 to 5 server generations, with a few hardware configurations per generation, are hosted at the same time on a data center [29]; and this hardware heterogeneity leads to an important variability in terms of server performance [28].

Contribution.
In this chapter, we cover the accounting of dynamic and static costs to VMs in heterogeneous data centers and introduce the following two tools:

- EPAVE, a power-aware attribution model for VMs taking into account the overall consumption of the data center hosting them
- PowerIndex, a profiling and estimation model for accounting the costs (in terms of utilization and execution time) of a VM when running on different nodes

Organization. This chapter is organized as follows. The EPAVE model for energy attribution is detailed in Section 4.2. Section 4.3 describes some practical use cases. Energy mapping with PowerIndex is described and evaluated in Section 4.4. We discuss the properties of both models in Section 4.5 and provide an outlook on the usage of EPAVE and PowerIndex in Section 4.6. Finally, Section 4.7 concludes the chapter.

4.2 Energy Attribution with EPAVE

The key idea of EPAVE is to attribute the data center's static and dynamic costs (C) to each VM, which can be then used as a basis for several use cases as described later. As costs we consider the total consumption (C_{total}) during the execution of a VM in the context of a data center.

$$C_{total} = C_{static} + C_{dynamic}$$

The static costs comprise the idle consumption of each node and the idle consumption of the routers as well as induced consumption of the entire data center (routers, air conditioning, power distribution units, etc.). To cover the entire data center the Power Usage Effectiveness (PUE) has become the industry-preferred metric for measuring infrastructure energy efficiency for data centers [59, 60]. It is defined as the ratio of total facilities energy to IT equipment energy:

$$PUE = \frac{Total\ Facility\ Energy}{IT\ Equipment\ Energy}$$

Therefore, we will use the PUE to account for the consumption part exterior to the IT equipment itself which is already taken into account. As outlined in the data center industry survey conducted by Uptime Institute Network (a user group of large data center owners and operators) [60], the adoption of PUE is rising worldwide, and its measurement and improvement are widely targeted by the 1,000 surveyed data center operators and IT practitioners. That is why we believe that the PUE metric is easily accessible for cloud providers for their data centers. From the PUE definition, for 1 Watt consumed by the IT equipment, the entire data center infrastructure consumes in fact $1 \times PUE$ Watts. Therefore, the static costs of a data center have to be multiplied by the data center's PUE:

$$C_{static} = \left(\sum^{\#nodes} C_{idle_{node}} + \sum^{\#routers} C_{idle_{router}} \right) \cdot PUE$$

The dynamic costs include the dynamic energy consumption part of the servers, routers and storage devices, which we can formulate as a weighted sum of the individual costs. The weights represent the resource usage of the current workloads, which can be between 0 and 1, where 1 means maximum utilization of the given resource and 0 means no utilization.

$$C_{dynamic} = \alpha \cdot C_{comp} + \beta \cdot C_{IO} + \gamma \cdot C_{net}$$

To attribute the overall costs to a single VM, we first need to distribute the idle costs in a fair and transparent manner. In many cases the idle costs (or idle power

consumption) are only divided by the number of VMs [25]. However, for energy-proportional accounting it is necessary to consider the size of a VM, and in particular, its number of virtual CPUs (vCPUs) as CPU is the most consuming device in a server [34]. Inspired by the VM types chosen by Amazon we will differentiate VMs by the number of their assigned virtual CPUs. In addition we want to take into account heterogeneous data centers.

The dynamic costs are determined when the VM finishes by using the real resource utilization of the physical resources. The total costs are limited to the static costs as the lower bound, and on the maximum consumption as the upper bound. Reporting these bounds to the user makes the final VM's costs predictable (bounded) and keeps the spirit of the pay-as-you-go manner although the dynamic part of the costs is in most cases smaller than the static parts (reflecting the reality of the power consumption of typical data center servers).

To sum up, the maximum costs of a VM grow with its size as shown by an illustrative example in Figure 4.1. In this experiment we were inspired by the Amazon VM sizes. The static costs, $C_{static}(VM)$, depend on the number of cores reserved by the VM, and the dynamic costs $C_{dynamic}(VM)$ on the actual usage (in our case performed by the stress command). In more details, we model the static and dynamic costs per VM as described in the following paragraphs.

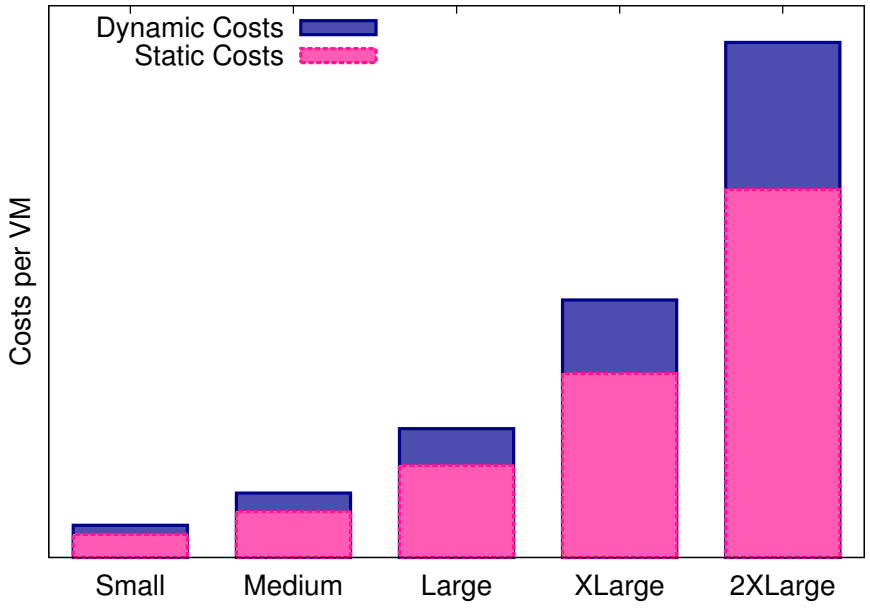

Figure 4.1: Example of maximum costs distribution among different types of VM.

4.2.1 Static Costs

As we want these costs to be static and independent from the hypervisor, we use a weighted averaged value of the idle power consumption of all the servers. This model is similar to the one currently in application at Amazon [61]: the costs are proportional to the number of vCPUs assigned to the VM:

$$C_{static}(VM) = \frac{\#vCPU(VM)}{\sum^{\#nodes} \#CPU(node)} \cdot C_{static}$$

4.2.2 Dynamic Costs

The dynamic costs are hardware and application dependent and require monitoring. According to [62] the acceptance of dynamic models is increased if the costs are limited by an upper bound. Indeed, a VM cannot exceed the physical resources allocated to it (CPU, RAM and disk mainly), so the upper bound can be determined for each type of VM over each type of physical node. The actual dynamic costs per VM will be in the range of 0 (idle) and the maximum resource usage. The challenge is to attribute the maximum dynamic costs to a VM. In general the dynamic costs of a VM are the measured or estimated energy consumption (E), which is the integral of the power consumption (P) measured/estimated per time unit (T).

$$C_{dynamic}(VM) = E(VM) = \int_0^T P(VM)\ dt$$

In general, a VM cannot consume more than the maximum dynamic costs of the entire server ($C_{dynamic}$). If the VM is co-located with other VMs it is necessary to split up the dynamic costs. Here, we use a very simple model to define an approximate upper bound for the costs of a VM, by using the number of cores the VM got assigned as a basis. Note that the focus on the number of vCPUs (ignoring the disk and network) is chosen because the number of vCPUs usually differentiates VM sizes offered by cloud providers. Additionally, the CPU is one of the highest power consumers on a node.

$$0 \leq C_{dynamic}(VM) \leq \frac{\#vCPU(VM)}{\#CPU(node)} \cdot C_{dynamic}(node)$$

An alternative would be to use a software power estimation model that is capable of attributing the dynamic costs to a VM, such as VMeter [25], or BitWatts [7].

4.3 Use Cases

In this section, we showcase how to calculate $C_{static}(VM)$ and $C_{dynamic}(VM)$ based on real-world experiments. Based on the real data we can use EPAVE to estimate the costs of different use cases. For the experiments, we rely on selected nodes from the Grid'5000 cluster to which powermeters are attached [63]. Specifically, we performed the experiments on two kinds of nodes, *Taurus* and *Sagittaire*, whose characteristics are specified as number 9 and 10 in Table 2.1 in Chapter 2. There are 16 Taurus and 79 Sagittaire nodes available. We further consider different sizes of VMs, which are inspired by the Amazon instances and shown in Table 4.1.

Homogeneous Setup

Figure 4.2 presents the costs $C_{total}(VM)$ for a homogeneous cluster with *Taurus* servers with 12 cores each. Their average idle power consumption is 95W per server. In a real setup the calculations need to include network costs and PUE, hence we need to add the costs for a number of switches (approx. 350W each) and multiply by the PUE (e.g., 1.22). In this specific example, we demonstrate the cost models based on the idle power of the servers as a matter of simplification for the calculations. The static costs per core are easy to compute: $95/12 = 7.92$. The dynamic part represents the maximal achievable dynamic power consumption when running the *stress* command. Together these costs represent the upper bound of costs per VM. We can see that the static costs increase proportional to the number of cores assigned to the VM and two VMs having together 12 cores will reach the same static costs as the machine itself. Hence, in a homogeneous setup EPAVE would fall back to a trivial model, where only the upper bound of costs $C_{total}(VM)$ have to be reported.

Type	Medium	Large	XLarge	2XLarge
Number of cores	1	2	4	8

Table 4.1: VM types.

Heterogeneous Setup

If we switch to a heterogeneous use case (as shown in Figure 4.3) and run again the *stress* command, the static costs are not proportional anymore to the number of virtual cores as the idle power of the machines might be unbalanced. We showcase the unbalanced scenario with experiments performed on two different kinds of servers,

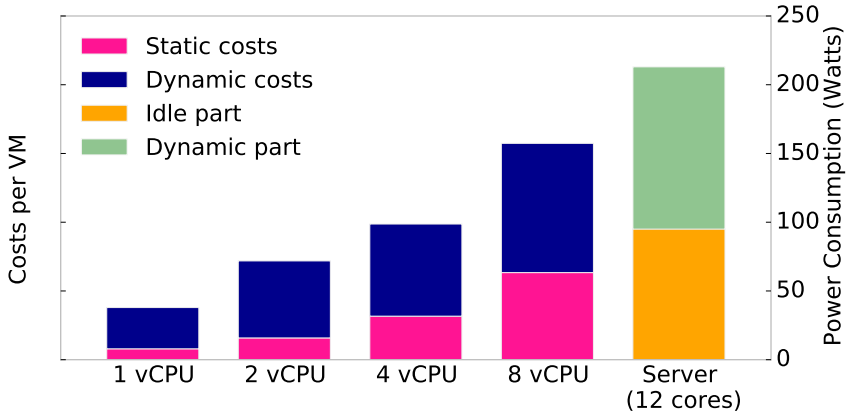

Figure 4.2: Example of maximum cost distribution among different types of VM for a homogeneous cluster.

whose characteristics are summarized as number 9 and 10 in Table 2.1 in Chapter 2. The two clusters are heterogeneous in terms of server architecture, but also in terms of number of nodes, and number of cores per node. The idle power consumption represents the average power consumption of a server over the entire cluster.
In this use case we can calculate the static costs for a one-vCPU VM as follows:

$$C_{static}(VM) = \frac{1}{\sum^{\#nodes} \#CPU} \cdot \sum^{\#nodes} C_{idle_{node}}$$
$$= \frac{16 \cdot 95 + 79 \cdot 215}{16 \cdot 12 + 79 \cdot 2} = 52.87$$

These costs are more than 6 times higher than in the homogeneous case with only *Taurus* nodes, but they represent half the costs of a cluster with only *Sagittaire* nodes. Therefore, heterogeneity among nodes leads to average static costs per virtual CPU which can be far from the costs per cluster. However, this is a healthy property of EPAVE: in order to cover the real energy costs with this accounting, the cloud provider has to favor the utilization of the most energy-efficient servers. To provide incentives for clients to use the most energy-efficient setup, we later introduce PowerIndex.

Underutilization of Reserved Resources

To show the applicability of our models, we performed experiments using real-world applications on a *Taurus* node. We installed Hadoop Yarn [64] on each of the nodes and ran *sort* and *wordcount* from the HiBench [65] benchmark suite. We run the

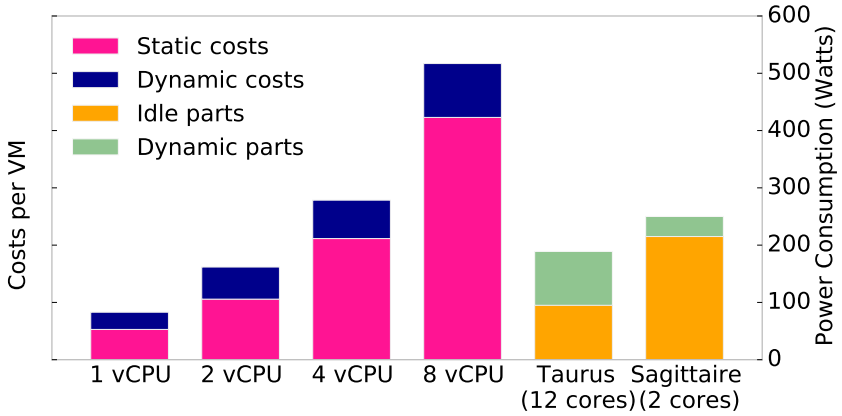

Figure 4.3: Example of maximum cost distribution among different types of VM for a heterogeneous cluster with unbalanced idle power for the server architectures.

workloads within a VM to be able to limit the number of cores they use in total. We started the VM once with only a single core, and once with all cores available. This experiment is the basis for three use cases, where we want to showcase the effects of underutilization of reservations. Note that the dynamic costs are always measured in real experiments and the static costs are predetermined based on the idle power consumption.

The workloads have different power consumption patterns as shown by the example of the *wordcount* workload executed on all available cores in Figure 4.4. The idle power of the *Taurus* nodes is 95W. We also know the maximum total power of 220W, and 125W as basis for $C_{dynamic}$ for all reserved cores without idle power. These values can be predetermined and have to be collected only once per architecture. The actual dynamic costs of the workload vary between 0 and 100W over a runtime of 200 seconds. This shows the necessity of considering energy rather than power consumption, as we need to provide models that reflect the actual usage of the VM over time.

If we consider the pay-as-you-go model as a basis, a VM would cost according to its size (i.e., resources reserved) and according to the time used. The same idea is followed by EPAVE, but we consider both static and dynamic energy as a basis of costs. As an example, for the single core experiment, we calculate $C_{total}(VM)$ according to our model and fill it with values from our experiments.

$$C_{total}(VM) = C_{idle} * ratio_{vCores} * runtime + C_{dynamic}$$

As shown in Figure 4.5 the static costs for using only a single core are smaller.

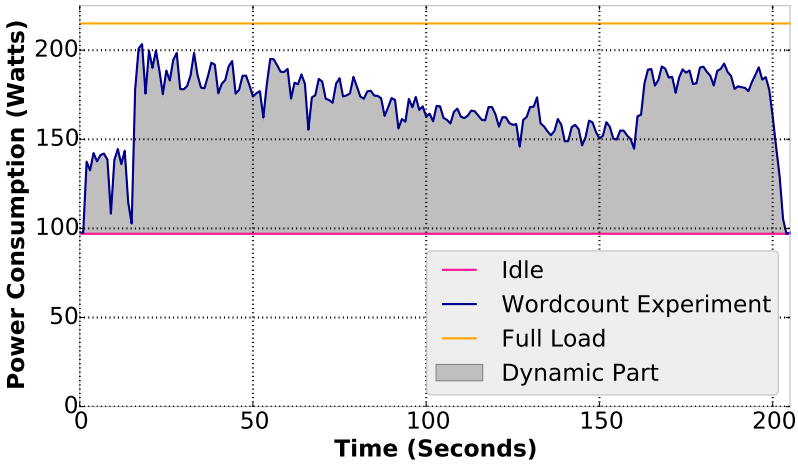

Figure 4.4: Power profile of the wordcount workload using all available cores.

However, because the single core is used for a longer time span, the dynamic costs are much higher leading to higher total costs than if all cores are used and reserved. Let us consider a use case where the workload is not optimized for parallelization, but still the reservation covers all of the cores. If a workload only uses a single core out of 12, the dynamic costs will not change in comparison to the former use case, however, the static costs are distributed among the number of cores served. Taking the dynamic costs of the former experiment as a basis, this would mean a significant increase in costs for the VM (see Figure 4.6). In an ideal case a user is encouraged to reserve resources according to the resources required and parallelization capabilities of the workload.

The runtimes of the former experiments are rather low and we assumed that the reservation for a VM ends with the end of the workload. However, in reality most VMs are reserved for a given time span. For instance, if we consider the default minimum reservation of VMs of around 20 minutes the cost distribution for the same workloads changes and the results are depicted in Figure 4.7. Hence, if we reserve all cores for 20 minutes but only use them for the first few minutes the static costs exceed the dynamic costs and the single core reservation is much more advantageous. With EPAVE it is possible for a user to identify such discrepancies and decide for what kind of reservation is useful. Another possibility is to use tools such as PowerIndex as introduced in Section 4.4 to provide insights on what costs should be expected when running in a heterogeneous setup.

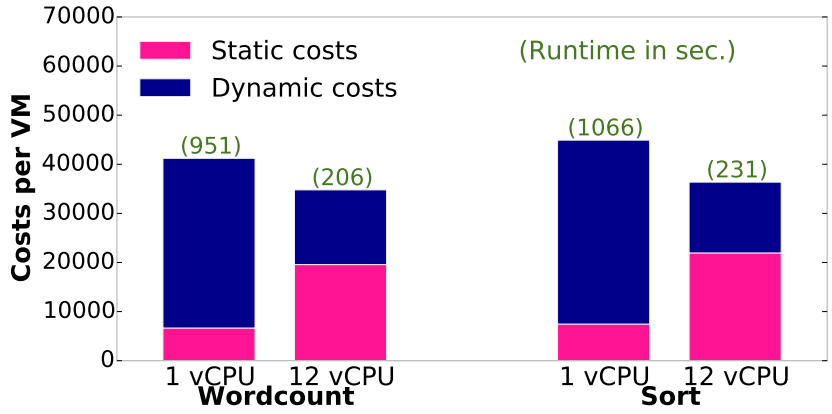

Figure 4.5: Costs of two parallel workloads with a reservation of one core and twelve cores.

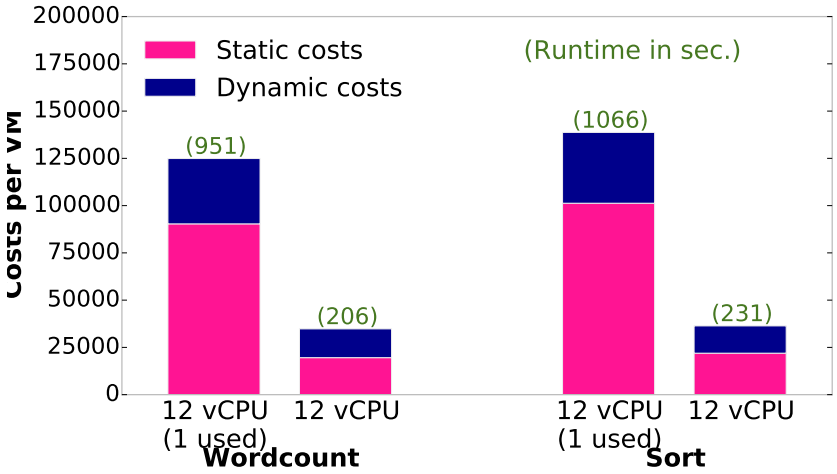

Figure 4.6: Costs of two workloads with underutilization of reserved cores.

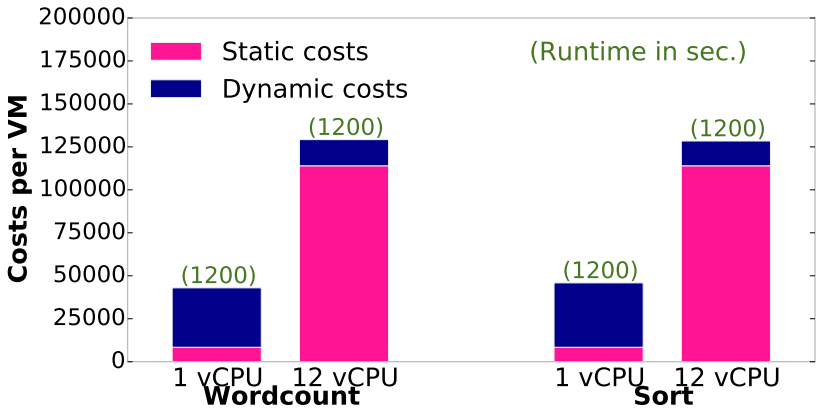

Figure 4.7: Costs of two workloads with predetermined reservation time of 20 minutes per VM.

4.4 Energy Mapping with PowerIndex

With EPAVE we are able to predict the costs of a VM in a coarse grained manner on a machine it should run on. For the final decision on where the user should run the VM, we introduce PowerIndex. PowerIndex is a tool to map different workload scenarios in a heterogeneous environment. Given a specific workload, PowerIndex provides an estimation of the potential energy consumption for execution on different types of machines. Usually, this would require to profile a given application on all available types of machines. This is obviously not efficient nor practical. The goal of PowerIndex is to be predictable and lightweight, only requiring minimal profiling effort. PowerIndex is a tool that is based on offline and online profiling, especially applicable for repetitive applications. The offline profiling is performed only once per machine type and allows us to build a reference power profile and mapping between a reference machine and all other machines. In the online phase a new workload is only profiled on the reference machine and the power and utilization mappings are used for predicting the VM's approximate costs on all other machines to perform proper scheduling decisions.

4.4.1 Offline Profiling

Our offline profiling approach consists of two main components: the Power Table and the Utilization Mapping.

Power Table

The *Power Table* stores information about the offline power consumption for each machine type (Machines M = m_1 , m_2, ..., m_m). To set up the Power Table, we execute different microbenchmarks consisting of CPU and disk-intensive workloads. The CPU-intensive microbenchmark comprises *stress* and *cpulimit*. We stress the CPU with different intervals of CPU load from 5 to 100 %, in steps of 5 %. For the execution of the workloads we want to cover both the user-space and the kernel-space utilization of the CPU and therefore cover *%usr* and *%sys* metrics. As a result we have a table of power consumption for *%usr* utilization intervals and for *%sys* intervals. The Power Table can be described formally for the profiling of a machine m_1:
Utilization U = $\{U_{usr}, U_{sys}\}$
Interval I = {(x,y),..}; $\forall x, y \in \mathbb{N}; x, y \in [0, 100]$
Power P = Watt $\subseteq \mathbb{N}$
Under the condition $\forall U_i \in U : U_i \subseteq I$
Define Power Table PT_{m1} (for *sys* and *usr*) for a machine m_1:
$\forall U_i \in U; \forall u_i \in U_i; f : U_i \times M \rightarrow P \; f(u_i, m_1) = p_{i,m1}$

Given a filled power table, function f maps the power consumption to a utilization interval, e.g., $f(u_i, m_1) = p_{i,m_1} = f((0, 20), m_1) = 10W$.

Utilization Mapping Table

When a workload runs on one machine, it will most likely not have the same utilization on another machine. Therefore, we need a basic understanding of how utilization u_{m1} translates to u_{m2}. Out of simplicity we provide the mapping between a reference machine $m1$ and all other machines and define the *Utilization Mapping Table*. We define a function ut_{sys} to map the system utilization from m_1 to the other machines. Given the utilization on the reference machine for a specific workload, we want to know what utilization to expect on the other machines.
Utilization $U \subseteq \mathbb{N}$, $\forall u_i \in U, u_i \in [0, 100]$
$\forall u_i \in U$, $\forall m_i \in M$
$ut_{sys} : U \rightarrow U$
$ut_{sys}(u_{i,m1}, m_i) = u_{i,m_i}$
The corresponding function for *usr* can be defined analogously.

4.4.2 Online Monitoring

The major part of our estimation is done in the offline profiling, which has to be only done once per machine type. The online monitoring is performed when a new workload arrives (that has not been logged already). PowerIndex requires that the new workload is run once for a configurable time on the reference machine and then uses the data from the offline tables to map the energy consumption on all other machines. The profiling time should be adapted to the expected workload. If the workload consists of repetitive phases, the profiling time should cover one or two phases.

4.4.3 Toy Example

To show the interaction between the offline and online profiling we consider a simple example. In a real data center, the workloads could be virtual machines or containers instead of simple workloads and the intervals would be chosen in a more fine-grained manner. Whenever a new unknown workload (w_1) arrives, we need to profile it on the reference machine and categorize it to obtain power values for each machine. If the workload w_1 is already known, we can just look up the required values in the database. PowerIndex holds the Utilization and Power Tables and we use them to get the final power consumption. Note that for this example we use a simplified set of Power and Utilization tables than described above. The utilization mapping assumes that machine m_2 has twice as many cores as machine m_1.

%usr	Power m_1 (W)	Power m_2 (W)
0	0	0
25	7.5	10
50	15	20
75	22.5	30
100	30	40

%sys	Power m_1 (W)	Power m_2 (W)
0	0	0
2	1	2
4	2	4
6	3	6
8	4	8
10	5	10

Table 4.2: Power Table for *%usr* and *%sys* utilization for machine m_1 and machine m_2.

With the given Power Tables and Utilization Mapping Tables we can start to estimate a simple workload, as described below.
Workloads W
$\forall w_i \in W : \exists! u_{sys} \in U_{sys} : p_{sys} = f(u_{sys}, m_1)$
$\forall w_i \in W : \exists! u_{usr} \in U_{usr} : p_{usr} = f(u_{usr}, m_1)$

%sys m_1	%sys m_2
0	0
2	1
4	2
6	3
8	4
10	5

%usr m_1	%usr m_2
0	0
25	12.5
50	25
75	37.5
100	50

Table 4.3: Utilization Mapping Table from reference machine m_1 to machine m_2 for *%usr* and *%sys*.

$p_{wi,mi} \in P$
$p_{idle} \in P$
$p_{wi,m1} = p_{idle} + p_{sys} + p_{usr}$

In this toy example, we assume a very short workload and that we monitor it entirely. At each point in execution, the power required for this workload on machine m_1 is the sum of the power required for the *sys* utilization, the *usr* utilization and the idle power. We assume that the reference machine has an idle power of 29W. The online profiling of workload w_1 on the reference machine m_1 results in the CPU utilization values as shown in Table 4.4.

Time (s)	%usr	%sys
1	50	6
2	25	6
3	25	6
4	100	0
5	100	0

Table 4.4: Utilization trace for the simple workload w_1 on machine m_1.

If we lookup the power for the utilization measured in Table 4.4 in the Power Tables of the machine m1, we can easily compute the energy by summing the power values per second:
$E_{m1} = (29+15+3)+(29+7.5+3)+(29+7.5+3)+(29+30+0)+(29+30+0) = 244J$
The execution of workload w_1 on machine m_1 would thus cost, in terms of energy, 244 Joule. To estimate the power on other machines, we need to map the measured utilization on m_1 to the utilization of the other machines using the Utilization Mapping Tables. Once we obtained the mapped utilization for machine m_i, we can get the power value from m_i's Power Table.
$\forall w_i \in W : \exists! u_{sys} \in U_{sys}$: measured utilization on m_1

$u_{sys,mi} = ut(u_{sys}, m_i)$: mapped utilization on m_i
$p_{sys} = f(u_{sys,mi})$
$\forall w_i \in W : \exists! u_{usr} \in U_{usr}$: measured utilization on m_1
$u_{usr,mi} = ut(u_{usr}, m_i)$: mapped utilization on m_i
$p_{usr} = f(u_{usr,mi})$
$p_{wi,mi} = p_{idle} + p_{sys} + p_{usr}$

This constructs the utilization trace as shown in Table 4.5.

Time (s)	%usr	%sys
1	25	3
2	12.5	3
3	12.5	3
4	50	0
5	50	0

Table 4.5: Expected utilization trace for the simple workload w_1 on machine m_2.

We can then lookup the power consumption for the mapped utilization values in the Power Tables for machine m_2 (Table 4.2). If we assume an idle power of 50W for machine m_2, we get the following energy consumption:
$E_{m2} = (50+10+3)+(50+5+3)+(50+5+3)+(50+20+0)+(50+20+0) = 319J$
We expect workload w_1 to cost 319 Joule on machine m_2. This toy example assumes that the execution time for workload w_1 is the same on machine m_1 and machine m_2. Otherwise, the execution time would be mapped similar to the utilization mapping presented in this toy example.

Real-World Use Case

In this subsection we consider a well known benchmark PARSEC on real hardware. We use again the *Taurus* and *Sagittaire* machines as defined in Table 2.1 in Chapter 2. These two types of machines have considerable differences in hardware (number of cores, CPU type, etc.) and therefore are very good examples for our experiments. We assume that the offline profiling, which is done once per architecture, has already been executed. Thus, the Power Tables and the Utilization Mapping Table are already available. We execute each benchmark for 20 seconds on the reference machine *Sagittaire*, and then look up the results from the offline profiling. We compare against the measured energy consumption with the power meters attached to the machines. Figure 4.8 shows the energy estimation for the Parsec benchmarks on the reference machine. We observe errors of less than 4%.

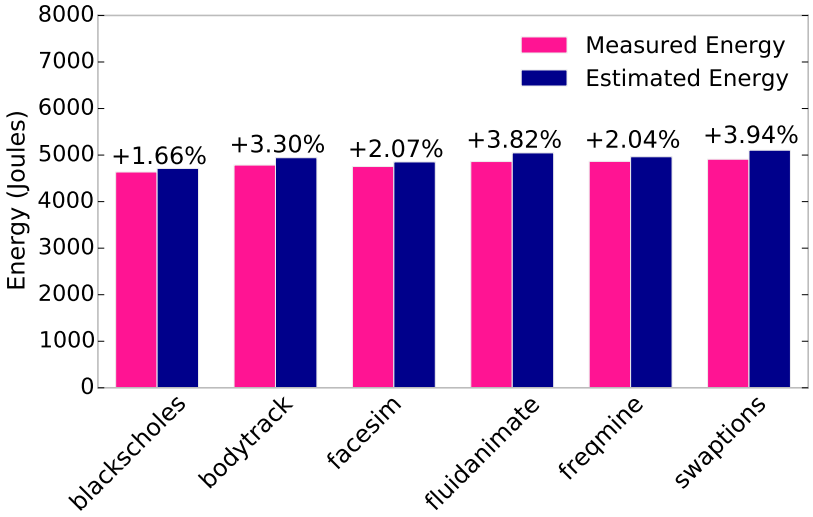

Figure 4.8: Energy estimation of the parsec workloads on the reference machine *Sagittaire*.

Figure 4.9 depicts the energy estimation on the *Taurus* machine. In this case we need to consult the Utilization Mapping Table and the Power Table. The energy estimation for the *Taurus* machine is based only on the values measured on the *Sagittaire* machine. We encounter an underestimation with a relative error that ranges between -4 % in the best case and -10 % in the worst case. The errors we encounter are acceptable in our use case. Errors up to 10 % are common in the related work of power and energy estimation [7]. In our case we can even accept errors that are slightly higher since we do not rely on high accuracy.

4.5 Discussion

EPAVE keeps the philosophy of the cloud: the pay-as-you-go model but based on energy consumption. The costs of a VM indeed depend on the physical resources reserved for it (static costs) and on the utilization made of these resources (dynamic costs). Moreover, the energy costs of a VM are predictable for the static part, and bounded (by the maximum costs as shown in Figures 4.2 and 4.3) and assessable through PowerIndex on heterogeneous nodes. Thus the user knows the maximal costs of the VM, and is able to estimate the real costs if the behavior of the running application and their energy consumption is known. With PowerIndex we extend the estimation of the dynamic costs on different machines by offering limited profiling on

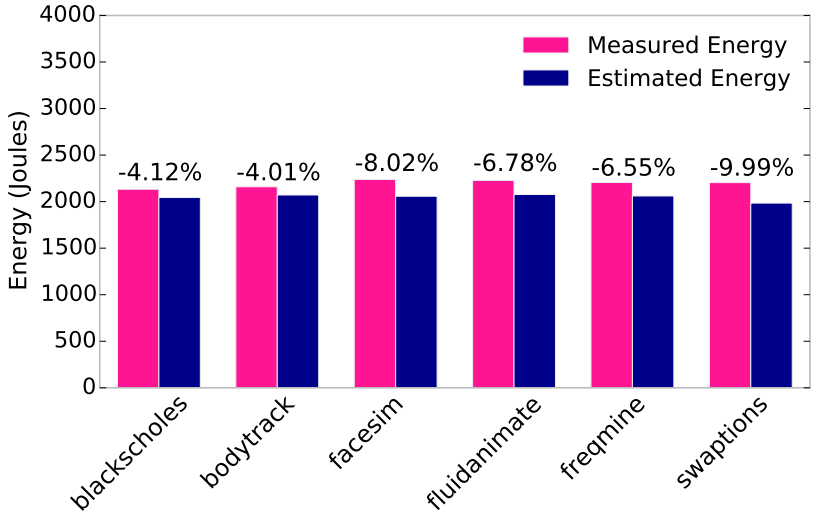

Figure 4.9: Energy estimation of the parsec workloads on the *Taurus* machine.

a reference machine.

However, EPAVE and PowerIndex together are not designed to account for the *real* cost of a given VM as it could be measured by external wattmeters during the entire lifetime of this VM. In this case, the cost of a VM would be influenced by cloud provider operations like VM migration or allocation of other VMs on the same host. This does not seem to be a desirable feature as it would reveal private information from the providers point of view. This is why EPAVE and PowerIndex are not based on this purely measurement technique and also why their goal is not to provide *real* measured costs but predictable, bounded, energy-proportional costs of a VM. These reflect the energy costs of an average VM with a given size hosted on a fixed cloud platform, similarly to what is done for pricing models [61].

EPAVE provides a complete view of the energy costs related to the hosting of virtual machines. Indeed, it does not only take into account server-related costs, but also the costs of the air conditioning, the networking devices, the power supplies, etc. That is why EPAVE can help the cloud provider to easily and fairly distribute the energy consumption of its entire infrastructure between the customers.

The computation of the energy costs determined by EPAVE relies, for the static side, on external power measurements (PUE, idle power of the servers), and for the dynamic side, on wattmeters or software-based tools. If these measured information are stored over time, the EPAVE energy costs can be re-computed later, thus becoming verifiable and auditable. PowerIndex only requires offline profiling once for each type

of machine, the online profiling relies on wattmeters or software-based estimation. Also values for PowerIndex can be recomputed after the short profiling phase.
EPAVE encourages users to dimension adequately their VMs. PowerIndex additionally helps to motivate the customers to go for the most efficient reservation in terms of energy. With the Utilization Mapping the case of underutilizing the VMs can be avoided. Indeed, if a user is asking for a 4-vCPU VM, but uses only 2 vCPUs, the two unused vCPUs will still be taken into account into the static costs – although their dynamic costs will be zero, even if the cloud provider is applying over-commitment of resources. Indeed, the dynamic costs are directly measured from the hardware, so all energy saving mechanisms employed by the user (e.g., energy-aware software) will be directly translated into a reduction of the dynamic costs of the VM. We assume here that the energy costs of a VM have somehow repercussions for the user (like a bonus-malus system, or monetary costs for VMs taking into account the energy).
In the case of heterogeneous servers, EPAVE in combination with PowerIndex encourage the cloud provider to use the most energy-efficient nodes. For instance, for the case described in Figure 4.3 with the *Taurus* cluster and the old *Sagittaire* cluster, a VM with 2 vCPUs will have static costs of 105.74 Watts. So, its static costs are bigger than the idle power consumption of a *Taurus* server, which is still able to host 5 more of such VMs. However, this VM's static costs are nearly twice smaller than the idle power consumption of a *Sagittaire* server which cannot host any additional VM.

4.6 Outlook

This section gives a non-exhaustive outlook on the application of EPAVE and PowerIndex, and more generically of the utilization of energy-aware cost models.

4.6.1 Pricing Models

EPAVE can serve as the basis for energy-aware pricing models. The static part is known at the beginning as it is defined by the VM type. For the dynamic part, the minimal bound is zero, and the maximal bound (for maximal energy consumption) can be provided to the user before the purchase. Reporting these bounds to the user makes costs per VM predictable (bounded) and keeps the spirit of the pay-as-you-go model because the dynamic part of the costs is in most cases smaller than the static parts (reflecting the reality of the power consumption of typical data center servers).

4.6.2 SLA with Renewable Energy Sources

Service Level Agreements (SLAs) provide quantified guarantees to the users concerning quality of service on the reserved VMs. In [66], the authors define green SLA: an explicit SLA for the percentage of renewable energy used to run the clients' workloads. In [67], the terms of green SLAs include also the energy costs of networking devices and virtual links between VMs. The green SLA is negotiated between the IaaS provider and each client depending on its needs. Such an SLA requires to have quantifiable green cloud services [66]. That is to say, the provider has to know the energy consumption of each VM and the electricity mix employed by the data centers. EPAVE can be used here to determine the energy budget spent by the VMs of a given user, and thus, to deduce the amount of green energy required for the cloud provider in order to fulfill the SLA conditions for this user.

4.6.3 User-Oriented Utilization

On the user side, EPAVE can be used as an energy cost metric in order to evaluate the energy efficiency of a given application running on given VM configuration. This metric can be used in combination with the classical metrics (duration, performance, QoS, etc.). By extrapolation, EPAVE can serve as a basis for a cost-benefit analysis including energy costs. Similarly, PowerIndex can be used for comparing different VM configurations for a given application, and thus determine the desirable trade-off between QoS and energy consumption.
Combined with energy-aware pricing models on the cloud providers side, EPAVE and PowerIndex can be an energy-aware incentive motivation. Energy-efficient users can be rewarded on the basis of their energy cost if they actively act towards its reduction. On the contrary, users can have an energy quota for running their VMs, which can be set by the provider or by the energy-aware user herself.
The application of EPAVE and PowerIndex described here are in particular possible because we do not only consider only the dynamic costs, and therefore, underutilization cases are penalized, as shown in Section 4.3. Finally, the utilization of EPAVE and PowerIndex simply display the energy costs of VMs could help raising energy-awareness of users.

4.6.4 Open Questions

EPAVE and PowerIndex leave some questions, which will be the subject of future work. In particular, EPAVE does not account for energy-saving techniques employed

by cloud providers, like switching off idle nodes. Therefore, it cannot be used to measure the energy efficiency of cloud facilities. Overcommitment is a classical technique employed by IaaS providers in order to decrease resource under-utilization and to maximize profit. EPAVE does not take this into account. PowerIndex relies on profiling parts of an application to make assumptions on costs on different machines. If the workload is very diverse in terms of power consumption, the prediction might be inaccurate. However, in combination with EPAVE the upper bounds of dynamic costs are known and can lower the risk of misplacement of workloads. This chapter presents our first attempt to build a reliable and intuitive model for energy accounting per VM in a heterogeneous cloud infrastructure. We hope this work will start paving the road towards energy-aware clouds.

4.7 Summary

In this chapter we introduced EPAVE, a model for predictable and transparent energy cost attribution per user. EPAVE is designed for simple usage, trying to keep the effort as limited as possible. The static costs comprise the PUE, which is already available in many data centers. The remaining static costs only have to be derived once. The only thing that requires constant monitoring are the dynamic costs, whereas the maximum dynamic costs can be pre-determined. In our experiments the actual dynamic costs are measured with a wattmeter as the nodes were used in a single-user mode. For multi-tenant usage a more fine-grained monitoring is required, such as provided by BitWatts [7] that additionally does not require a wattmeter (except for the model building phase). As a help to dimension VMs we also introduce PowerIndex, a simple profiling tool that allows to display the costs of a workload running on different types of machines. This tool comprises an offline profiling phase required only once. The online monitoring phase is done once per new application, and only performed for a limited amount of time.

5 BitWatts: Process-Level Power Estimation in VM-based Systems

This chapter introduces BitWatts, a middleware toolkit for building software-defined power meters. It addresses the host/virtualization layer of the stack presented in Chapter 1 by providing fine-grained power estimation at a process-level. BitWatts estimates the power consumption on physical machines as well as in virtualized environments.

5.1 Introduction

Context. Energy-efficient computing is becoming increasingly important. Among the reasons, one can mention the massive consumption of large data centers, estimated to account for about 2% of global greenhouse gas and some of which consume as much as 180,000 homes [1, 68]. This trend, combined with environmental concerns makes energy efficiency a prime technological and societal challenge.
Researchers and operators have been proposing solutions to increase energy efficiency at all levels, from application to runtime and to hardware. As surveyed by Orgérie et al. [34], examples include methods for energy-based task scheduling, energy-efficient software, dynamic frequency and voltage scaling, and energy-aware workload consolidation using virtualization. Virtualization offers environment and performance isolation and, hence, is the basis for many data center and cloud management frameworks. In order to improve their energy efficiency, such cloud management frameworks need to know the resource requirements of the running entities.
For data center providers and users, it is particularly useful to identify which applications are the largest power consumers. However, physical power meters and components with embedded energy sensors are often missing, and they require significant investment and efforts to be deployed a posteriori in a data center. Additionally, these hardware facilities usually only provide system-level or device-level granularity. Hence, software-based power estimation is becoming an economical alternative [34].

Power estimation is relatively accurate when one has full control over the underlying hardware and detailed knowledge of its properties. It typically works by sampling the activity of applications and measuring the power consumption of the whole system using hardware-specific probes.
In virtualized environments, one does not have direct access to the physical CPUs and one can only observe the processor emulated by the *virtual machine*'s (VM) hypervisor. Furthermore, the physical resources available to the emulated CPU may change dynamically as a result of VM scheduling—a VM may run alone on some physical core(s) for some time and later compete with other VMs—or even migrate to another host.
Current approaches providing power estimation remain poorly adapted to virtualized environments and do not provide acceptable measures. The few existing approaches either consider the VM as a black-box running a single application [18, 24], or they require extensions to the hypervisor or to the host and guest operating systems for being operational [27, 21].

Motivating scenarios. The introduction of fine-grained power monitoring within virtualized environments opens up for new scenarios.
Platform-as-a-service (PaaS) infrastructures such as Google App Engine allow developers to create programs that run in sandbox mode [69]. Request and database handling is performed outside of an application in separate tasks. To isolate which application draws the most power, it is necessary to cover each individual process. This does not only allow for new power-aware pricing models, but also helps improve energy proportionality mechanisms.
In cases of dedicated cloud offers[1] or nested virtualization, such as proposed by Ben-Yehuda et al. [70], an *infrastructure-as-a-service* (IaaS) provider could offer user-controlled hypervisors within a VM. This allows cloud users to run their favorite types of hypervisors and accompanying VMs. However, the management of VMs in such environments can become deeply complex and, with current solutions, it is impossible to monitor the power consumption of a single VM at the highest level of nesting. This prevents typical tasks, such as resource and power provisioning. Such use cases therefore require a flexible solution that can operate on local, nested, and distributed levels without extra efforts.
More specifically, consider a distributed setup with nested virtualization in which we would like to track the power consumption per VM and per user in order to apply

[1] https://www.ovh.com/ca/en/dedicated-cloud, accessed on 17.07.2016

power-aware pricing. Such a setup is illustrated in Figure 5.1. One VM per user can be initially started on each node, and the user can subsequently launch additional VMs running multiple processes within the provided environment. In such settings, it is desirable to be able to monitor the power consumption of each of the user's processes and VMs separately. Furthermore, as a user might operate on multiple nodes, distributed monitoring of the energy consumption of all his processes is also instrumental to determine per-user energy consumption for the pricing model. Our BitWatts system, which we present in this chapter, provides such facilities.

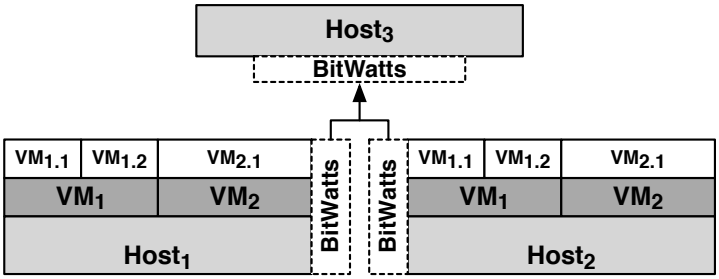

Figure 5.1: Example for BitWatts acting in a multi-tenant virtual environment.

Contributions. In this chapter, we propose BitWatts, a middleware solution to estimate the power consumption of software processes running in virtualized environments.[2] While BitWatts is a modular framework that can accommodate different power models (including *running average power limit* (RAPL) probes and power meters), we propose a process-level power model, which is application-agnostic and accounts for virtualization—*i.e.*, for emulated cores within a VM—and for the power-aware extensions of modern processors, notably HyperThreading and *dynamic voltage and frequency scaling* (DVFS). In our software solution, we expose power probes from the host *operating system* (OS) to the guest OS so that BitWatts can estimate the power consumption of processes running within a VM. In addition, this design can operate in distributed settings, with consumption information transmitted over high-performance publish/subscribe middleware.

We have implemented BitWatts in Scala, as an extension of the PowerAPI actor toolkit.[3]

We ran extensive experiments with several workloads on various computer settings, and we assessed the accuracy of BitWatts by comparing its output to physical measurements performed with a power meter. Results indicate that BitWatts provides

[2]Available as open source at: http://bitwatts.powerapi.org
[3]PowerAPI (AGPL): http://powerapi.org

trustworthy power estimation within a few per cent of actual measures when configured with the appropriate power model for the underlying hardware. We describe the design of such a CPU power model, which is application-agnostic, supports both CPU- and memory-intensive workloads and is processor-aware, including *multi-cores*, *HyperThreading*, *dynamic scaling*, and *dynamic overclocking* features.
In a typical server, the major power consumer is the CPU [34], covering at least one third of the overall power consumption. Hence, like other studies [17, 26, 71, 20], our power model focuses on processor consumption and accurately monitors applications that are CPU- and memory-intensive. For disk-intensive workloads, we need further studies and finer-grained models since two hard disks, even of the same model and making, have different power consumption patterns [72]. Therefore, we selected our benchmarks in such a way that the additional power possibly consumed by the disk is negligible. Studies in data centers [34] showed that network I/O (in the case of Ethernet) is not impacting the power consumption as the difference between idle and fully utilized links is negligible.
BitWatts is a collaboration between the University of Neuchâtel and the University of Lille / INRIA Lille. This chapter focuses on the virtualization aspects of the paper presented at EuroSYS 2015 [7], which are the contributions of the author of this thesis. Some contents regarding the details of the CPU power modelling are left out and can be found in the paper.

Outline. The rest of this chapter is organized as follows. We first compare our approach to the related work (presented in Chapter 2) in Section 5.2. We then introduce the general principle and the architecture of BitWatts in Section 5.3 and describe the power models in Section 5.4. We provide an in-depth evaluation in Section 5.5 and finally, conclude in Section 5.6.

5.2 Comparison to Related Work

As a summary of the current state of practice as shown in Chapter 2, the existing CPU power models found in the literature cannot be reproduced because *i)* the details of the selected counters are not provided [16] or sufficiently documented [20], *ii)* they are tailored to a specific processor architecture (including a limited set of power-aware features) [71], or *iii)* they build on private workloads that cannot be reused to assess alternative power models [20]. BitWatts differs from the state-of-the-art by providing an open source implementation of the proposed toolkit and builds on

standard counters and benchmarks (*e.g.*, `stress`, PARSEC, SPECjbb) to provide an open testbed for CPU power models.
More specifically speaking, one could note that the literature has been mostly focusing on the definition of power models for physical machines by trying to cover some of the power-aware features of multi-core processors. Nevertheless, the existing approaches consider each feature separately and, to the best of our knowledge, none of them provide a CPU power model that accounts for all of these features in the context of multi-core systems that run several applications concurrently.
With regard to the power consumption of VMs, state-of-the-art solutions provide no or limited support for fine-grained monitoring of applications running within a VM. The few existing approaches either consider the VM as a black-box running a single application, or they require extensions to the hypervisor or to the host and guest operating systems for being operational.
In this chapter, we therefore propose to tackle both challenges by reporting on the design of software-defined power meters that can run both on the host and in a VM. In particular, on the host, we propose a first configuration of a software-defined power meter that builds on a new CPU power model that accounts for common power-aware features of multi-core processors to deliver accurate power estimation at the granularity of a software process. In the VM, we introduce a second configuration of a software-defined power meter that connects to the host configuration in order to distribute the power consumption of VM instances between the hosted applications. The proposed configuration can even be extended to consider distributed power monitoring scenarios involving application components spread across several host machines.
Unlike existing approaches found in the literature, the CPU power models we describe *i)* are application-agnostic, *ii)* are processor-aware, and *iii)* scale with the number of software processes to be monitored concurrently. We assess these properties by reporting on the errors observed for both CPU-intensive and memory-intensive applications provided by acknowledged benchmarks.

5.3 Software-defined Power Meters

Power estimation of processes running in virtualized environments is not a trivial task, since several factors have to be considered. In particular assumption, such as the presence of a single application running in a single VM on a single core, do not hold anymore. One has to deal with complex scenarios with a number of VMs that

may exceed the number of physical cores and several applications that run within each VM. To cope with these different dimensions of scaling, we designed and implemented the BitWatts middleware framework as a modular solution to build software-defined power meters. In the rest of this section, we give a high-level overview of its architecture and implementation.

5.3.1 Architecture Overview

BitWatts relies on a multi-tier architecture, depicted in Figure 5.2, that shares the power consumption of the VMs running on the host to the application processes running within the VM. Since the VM does not have direct access to the hardware, we use a fast communication interface to connect instances of BitWatts running on the host and in the VMs. Similarly, BitWatts also supports communication across machines using publish/subscribe communication channels to report consolidated power estimation values of distributed applications spanning multiple nodes (*e.g.*, in a cluster).

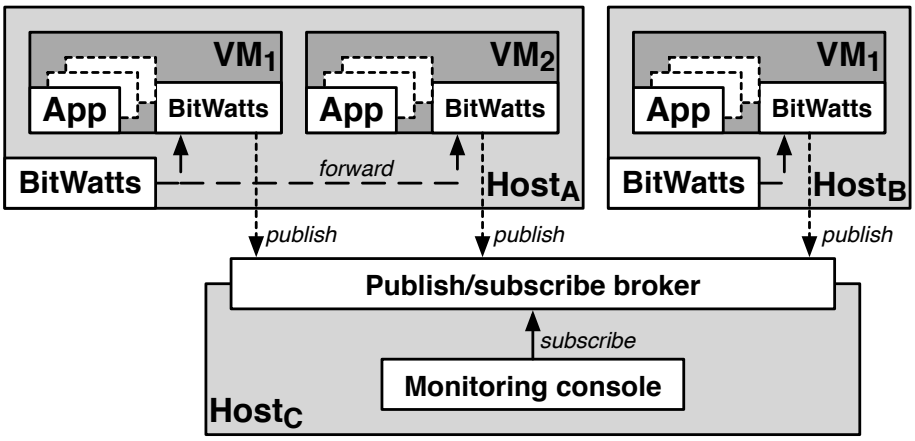

Figure 5.2: BitWatts middleware architecture.

5.3.2 Power Meter Middleware Toolkit

We built BitWatts as a modular middleware solution to assemble software-defined power meters.

Software-defined power meters are customizable solutions that can deliver power consumption reports at various frequencies and granularity, depending on the power

monitoring requirements. In particular, this chapter focuses on per-second process-level monitoring in order to closely monitor the activity of an application running on a system.
Our solution builds on the PowerAPI toolkit [73], which adopts the actor programming model as a solution that can scale with the frequency and the number of applications to be monitored. The software components of BitWatts are therefore implemented as actors, which can process millions of messages per second, a key property for supporting real-time power estimation. More specifically, the PowerAPI toolkit identifies four types of actors that are reused and extended in BitWatts:

- **Sensor** connects the software-defined power meters to the underlying system in order to collect raw measurements of system activity. Raw measurements can be coarse-grained power consumption reported by third-party power meters and embedded probes, or CPU activity statistics as delivered by the process file system (`ProcFS`). Sensors are triggered according to the requested monitoring frequency and forward raw measurements to the appropriate formula.

- **Formula** uses the raw measurements received from the sensor to compute a power estimation. A formula therefore implements a specific power model [24, 18, 19] to convert raw measurements into power consumption. The granularity of the power consumption reported by the formula (machine, core, process) depends on the granularity of the measurements forwarded by the sensors.

- **Aggregator** is in charge of aggregating power consumption, according to a specific dimension like the *process identifier*, to compute the energy consumption, or *timestamp*, to group the power consumption of several applications.

- **Reporter** finally formats the power consumption produced by the formula or the aggregator into a suitable format. Such reports can be provided for instance via a Web interfaces or a virtual file system (*e.g.*, based on FUSE).

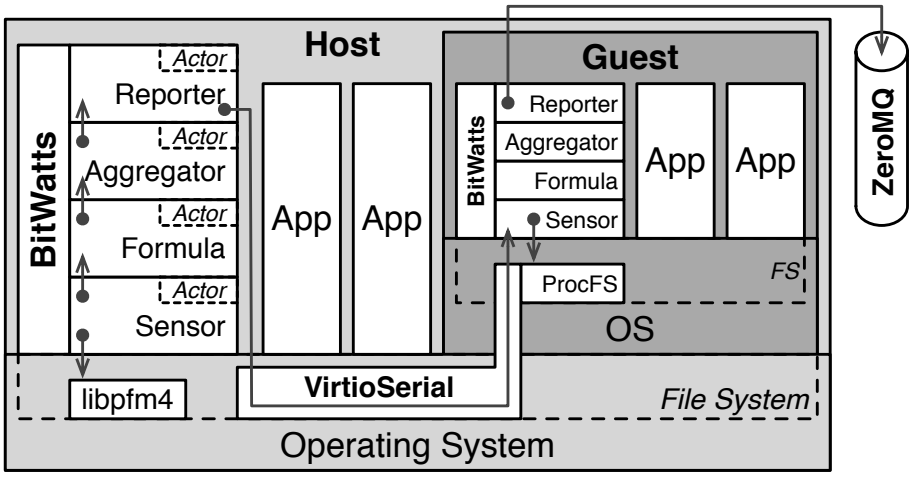

Figure 5.3: BitWatts middleware implementation.

As the BitWatts middleware framework supports process estimation in VM-based systems, implementations of the sensor, formula, and reporter actors are assembled in different configurations on the hosts as well as in the VMs (see Figure 5.3).
Additionally, to improve the accuracy of state-of-the-art power estimation, we deliver a new power model that builds upon a libpfm[4] sensor actor on the host to collect the hardware performance counters associated to the monitored VM process. The formula actors consume the measurements collected on the host by this libpfm sensor to estimate the power consumption of a given process. The resulting consumption measures are automatically published by two reporter actors through two different communication channels: VirtioSerial[5] and in a distributed setup also to ZeroMQ.[6] The data forwarded through these channels is consumed by sensor actors.
The BitWatts middleware framework therefore provides an exhaustive toolkit to assemble software-defined power meters on demand. The results reported in the following sections are notably based on a variety of software-defined power meters built with BitWatts to: monitor coarse-grained power consumption using a third-party power meter or RAPL probes, learn the power model of the processor, deliver process-level power consumption on the host, and report on fine-grained power consumption within the VMs.

[4]http://perfmon2.sourceforge.net, accessed on 17.07.2016
[5]http://www.fedoraproject.org/wiki/Features/VirtioSerial, accessed on 17.07.2016
[6]http://www.zeromq.org, accessed on 17.07.2016

5.3.3 Power Consumption Communication Channels

Exchanging data between instances of BitWatts requires two levels of communication. First, we need to exchange data between the host and the VM to estimate the power consumption of a process within the VM. Second, in a distributed setup, we want BitWatts to report the power estimation to another server, *e.g.*, to aggregate the data monitored on multiple physical or virtual nodes.

For the hierarchical communication between instances of BitWatts running on the host and a VM, a lightweight transport mechanism is required to exchange messages at a high rate while crossing the VM boundaries.

VirtioSerial is based on the file system and has been developed for the very purpose of inter-VM communication. It provides the performance required to reduce likelihood of synchronization errors of power measurements between host and virtual machine. The VirtioSerial hierarchical communication channel is implemented in BitWatts as a reporter actor on the host and a sensor actor in the VM (see Figure 5.3). Multiple instances of BitWatts are running concurrently: one in the host and one per VM. For the host, the VirtioSerial reporter communicates the power consumption of the VM process to the `virtio-pci` device. In the VM, the VirtioSerial sensor connects to the VirtioSerial port and reads power consumption reported by the host. The BitWatts formula uses these values to compute the process-level power consumption within the VM.

In a distributed setup, we need to communicate across machines, typically to aggregate the power measurements from distributed application components running on different VMs and hosts. Our distributed communication channel therefore consists of a publish/subscribe system using ZeroMQ. ZeroMQ is a networking API that supports complex messaging patterns and provides bindings for various programming languages while being lightweight. The key component of the publish/subscribe system is the broker. It forwards messages received from the distributed BitWatts instances to interested subscribers, for example loggers or the monitoring console (see Figure 5.2). Messages exchanged between BitWatts, the broker, and the subscribers are serialized using Apache Thrift,[7] an efficient interface definition language and binary serialization protocol.

[7] http://thrift.apache.org, accessed on 17.07.2016

5.4 Process-level Power Models

BitWatts relies on specific power models to estimate power consumption of individual processes. Per-process power estimation is a cornerstone to identifying the largest power consumers and to take informed decisions. In particular, we discuss in this section how such power models can be connected to support power estimation within a VM.

5.4.1 Multi-core CPU Power Model

To control energy consumption, CPUs rely on frequency scaling and power saving modes to adjust their performance according to computation requirements. In particular, the multi-core processors designed by Intel integrate the following features:

- **HyperThreading (HT)** is used on some processor generations (*e.g.*, Pentium IV, Xeon) to separate each core into two threads. The technology is based on the *simultaneous multi-threading* (SMT) principle, which allows the processor to seamlessly support *thread-level parallelism* (TLP) in hardware and share more effectively the available resources. Performance gains strongly depend on software parallelism, and for a single-threaded application it may be more effective to actually disable this technology.

- **SpeedStep (SS)** is Intel's implementation of *dynamic voltage/frequency scaling* (DVFS), which allows a processor to adjust its clock speed and run at different frequencies or voltages upon need. The OS can increase the frequency to quickly execute operations or reduce it to minimize dissipated power when the processor is under-utilized.

- **TurboBoost (TB)** can dynamically increase the processor frequency beyond the maximum bound, which can be greater than the *thermal design power* (TDP), for a limited period of time. It therefore allows the processor cores to execute more instructions by running faster. TurboBoost is however only activated when some specific conditions are met, notably related to the number of active cores and the current CPU temperature. It also depends on the OS, which may request to trigger it when some applications require additional performance.

Table 2.1 in Chapter 2 reports on the configuration of two families of Intel processors that exhibit different features and are used in our evaluation of BitWatts (number 1

and 11). The i3 has SpeedStep and Hyperthreading, whereas the Xeon has SpeedStep, HyperThreading and TurboBoost. Their internal complexities are reported by the *portable hardware locality* (hwloc)[8] software package and detailed in Figure 5.4. These two configurations differ by the number of cores and threads available as well as the CPU features (TurboBoost) that can be exploited by the operating system.

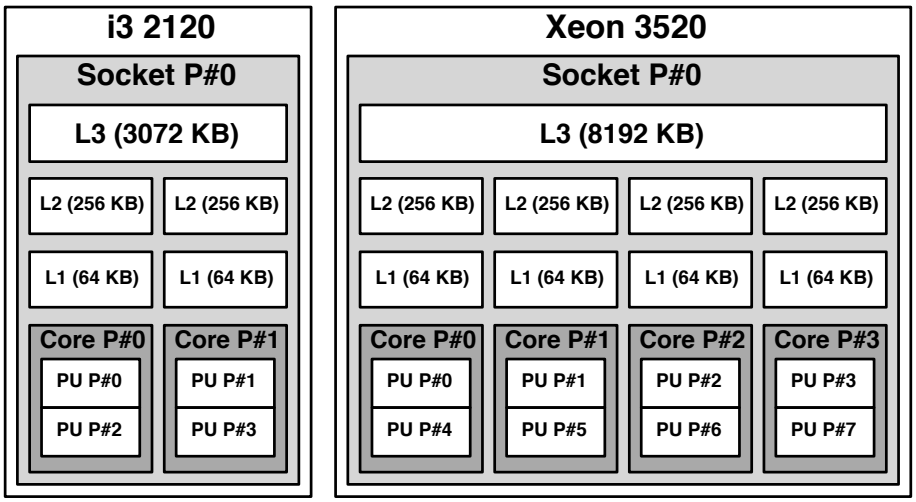

Figure 5.4: Core i3 and Xeon topologies.

CPU power model. The learning of the power model of multi-core processors requires the definition of workloads that carefully stress the various features it support as HyperThreading, SpeedStep and TurboBoost. Based on such workloads, power measurements and hardware performance counters, the power model was defined. For more details, please refer to the paper [7].

Power model assessment. First, to demonstrate that BitWatts is able to handle applications with diverse load, we start with a baseline experiment on the *i*3. We run the `stress` tool on a single core in combination with `cpulimit`. Every 30 seconds, the stress load is decreased by 10 %. In this experiment, we compare the results not only to PowerSpy, but also to *running average power limit* (RAPL) counters, which report CPU-package power consumption and are available on recent Intel processors (since the *Sandy Bridge* processor generations and hence on the *i*3). Furthermore, for this experiment, we set the CPU frequency to a fixed ratio of 1.6 GHz to avoid peaks in the power measurements of PowerSpy.

[8]http://www.open-mpi.org/projects/hwloc, accessed on 17.07.2016

Figure 5.5 shows the results of the workload executed on the host. We see that the RAPL counters follow the trend of the workload, but tend to overestimate the power consumption of a single CPU. Compared to RAPL, BitWatts provides power estimation that is much closer to PowerSpy that we consider as the ground truth. This indicates that BitWatts performs accurate sub-system estimation in various load scenarios, which is a prerequisite to be able to monitor virtual machines using a subset of the resources of a physical host.

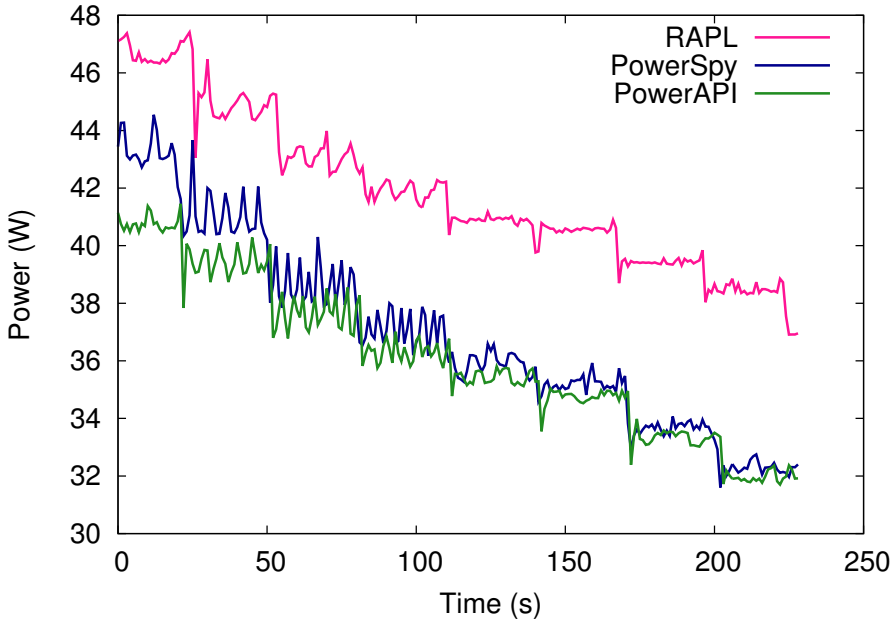

Figure 5.5: Decreasing load of stress on $i3$ in the host, compared to RAPL.

In the next scenario, we assess our power model for multi-threaded applications in comparison to PowerSpy. This comparison uses the well-known PARSEC [74] v2.1 benchmark suite, which includes many CPU-intensive workloads. This suite was designed to stress all the resources available on multi-core architectures. In particular, we report the power consumption of all the benchmarks available on two different configurations used in our tests. Figures 5.7 and 5.6 report the relative error between the measured and estimated power consumption (by aggregating the power consumption per process using P_{host}).

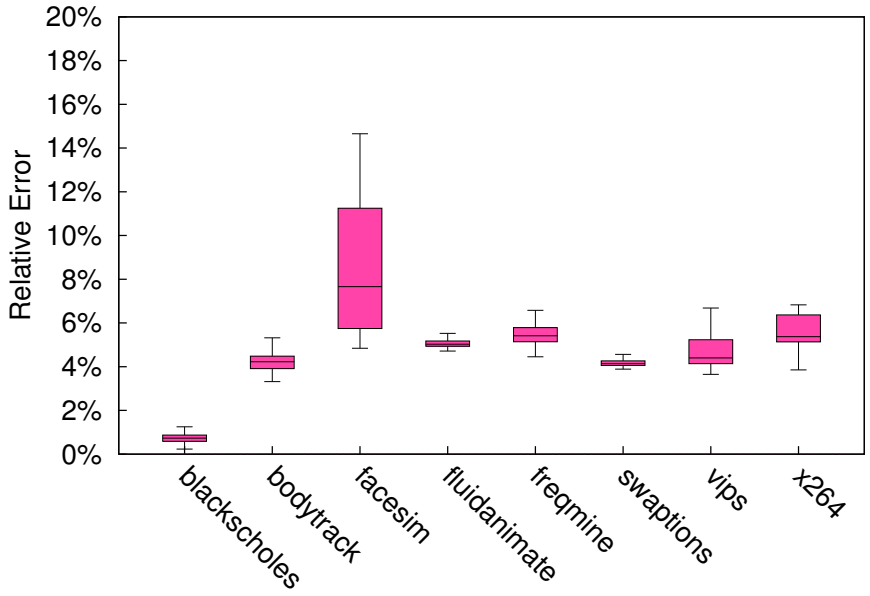

Figure 5.6: Relative error distribution of the PARSEC benchmarks on the i3 processor.

Even though PARSEC was not included as a workload during the sampling phase, one can observe that the estimation produced by our power models is close to the power measurements collected for the two different processor models considered. The closest method to ours, described in [19], adopts an iterative approach to minimize the error rate to at most 5 %. However, the key limitations of their approach are *i)* they only consider full usage of the cores, and *ii)* they rely on an application-specific model. Our solution is application-agnostic, supporting both CPU- and memory-intensive workloads, and are processor-aware, considering different models of CPUs including *multi-cores*, *HyperThreading*, *dynamic voltage/frequency scaling*, and *dynamic overclocking* features.

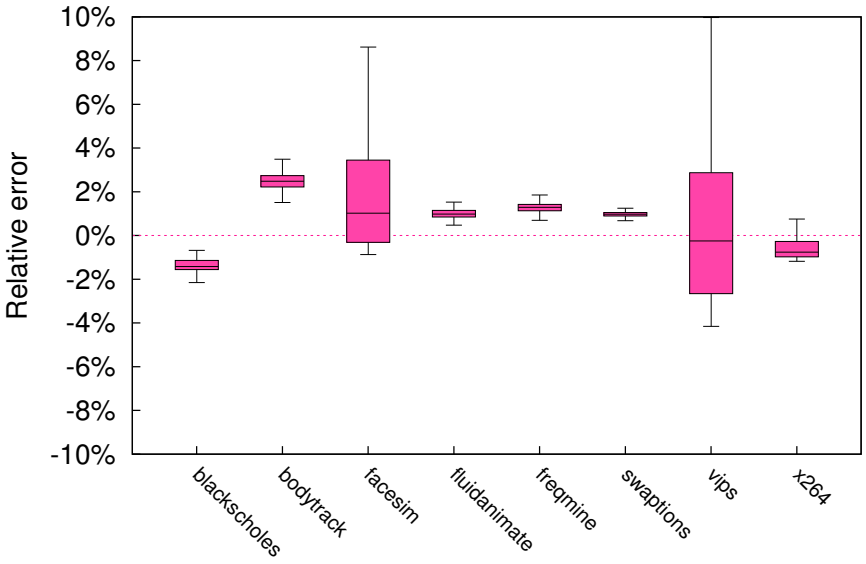

Figure 5.7: Relative error distribution of the PARSEC benchmarks on the Xeon processor.

Figure 5.8 illustrates the capability of estimating and isolating the power consumption of concurrent processes running on the same CPU. In particular, it shows how the power consumption of the Intel Xeon configuration is distributed between the idle power consumption and two benchmarks taken from the PARSEC suite (x264 and freqmine). Compared to physical measurements, when running at a frequency of 4 Hz (every 250 ms), our solution achieves a median error of 0.30 % with a maximal error of 9.73 %, thus competing with *post-mortem* analysis like [26].
Regarding the monitoring frequency, BitWatts is mostly limited by the frequency of the hardware and software sensors used to collect runtime metrics. In particular, BitWatts can report on the power consumption of software processes up to 40 Hz when connected to the PowerSpy, and up to 10 Hz when using the `libpfm4` library. However, by increasing the monitoring frequency one can observe that the stability of power consumption is affected, which does not help to properly identify the power consumption of the processes. Therefore, in the rest of the chapter, we configure BitWatts to report on the power consumption with a frequency of 1 Hz in order to smooth the reported values.

Additionally, Figure 5.8 reports on the power consumption of BitWatts during execution. The power consumption of 5.4 W on average demonstrates that our implementation of the power model has a reasonable footprint and is weakly impacted by the number of processes being monitored. This footprint acknowledges the design and the implementation of BitWatts as a scalable actor toolkit to build software-defined power meters.

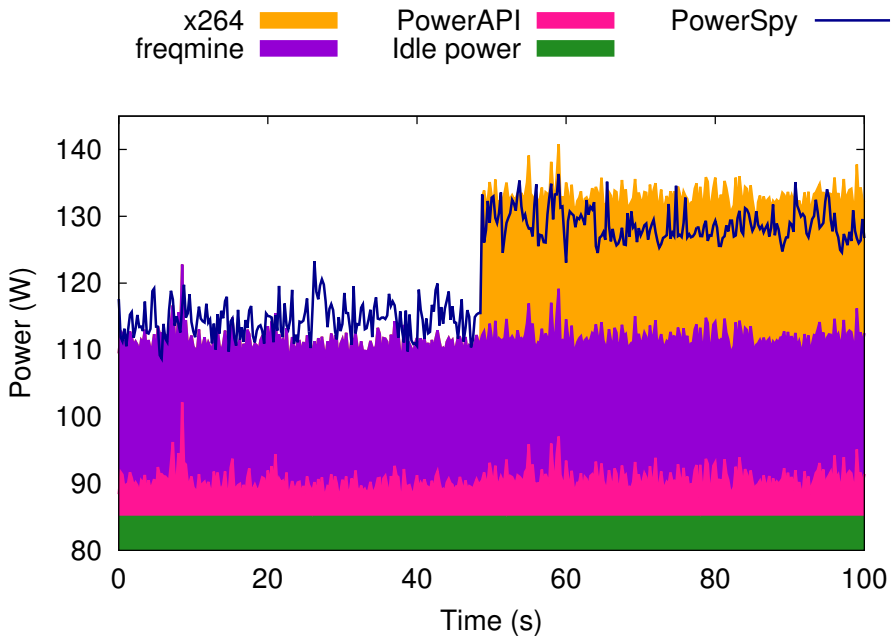

Figure 5.8: Process-level power consumption of BitWatts, x264, and freqmine on the Xeon processor.

Generality of the model. While the multi-core CPU power model proposed in this chapter is only assessed on Intel processors, the solution that we describe does not rely on any Intel-specific extensions. Indeed, our model considers processor features (HT, SS, TB) that are also available from other vendors. In particular, AMD processors also represent a target CPU architecture for our power model, but a limitation of the `libpfm4` library currently prevents BitWatts to access the `reference-cycles` to compute the current frequency. Once this barrier is lifted, we expect to be able to also demonstrate the validity of our model on AMD processors with results similar to those reported for Intel.

5.4.2 Virtual CPU Power Model

Unlike the architectures observed at the host level (see Figure 5.4), virtual CPUs tend to be simpler: they map physical cores to logical processors (sockets) and typically do not support any SS/HT/TB features, as illustrated in Figure 5.9. Hence, when pinning a single-core VM on a physical core of the host, the power consumption of a process running in the VM is proportional to the CPU utilization of the VM on the host.

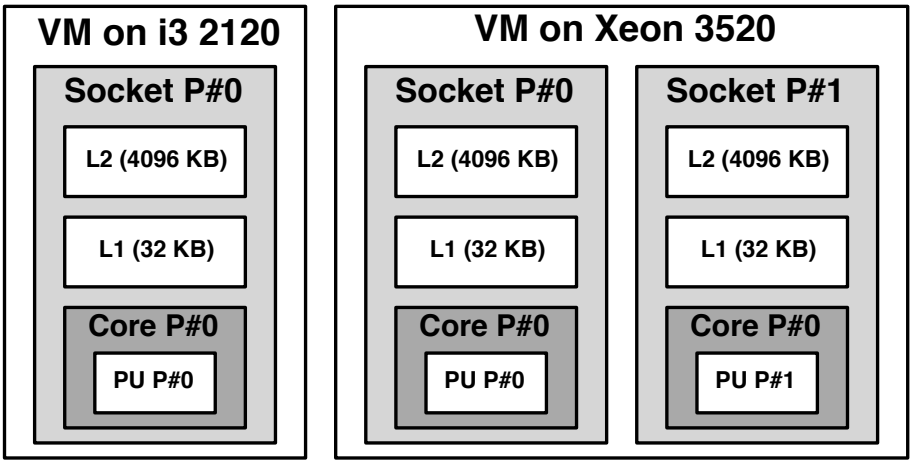

Figure 5.9: Core i3 and Xeon VM topologies.

To estimate the power consumption of an application running in the VM $P_{vm}(app)$, we need therefore to know the consumption of the VM process $P_{cpu}(vm)$ on the host machine, as well as the CPU utilization of the application $U_{vm}(app)$ relatively to the other applications running in the VM $U_{vm}(total)$:

$$P_{vm}(app) = P_{cpu}(f, uc^1_{vm}...uc^N_{vm}) \cdot \frac{U_{vm}(app)}{U_{vm}(total)}.$$

BitWatts uses a sensor in the VM to monitor the utilization of the application under observation and of all processes running in the VM. Another sensor gathers information about the power consumption of the VM forwarded by the host. The formula then computes the power consumption based on the model and forwards the results to a reporter. Note that this reporter can be used to implement distributed energy monitoring scenarios using publish/subscribe middleware, as described in the next section.

5.5 Evaluation of BitWatts

In this section, we report on the experimental results we obtained for BitWatts. In particular, we show that accurate host power estimation and efficient communication with the VM are necessary to support power estimation in realistic virtualized environments.

5.5.1 Experimental Setup

The experimental setup consists of two types of servers (i3 and Xeon) with different hardware characteristics. For the distributed setups, we use three identical servers of type i3.
We rely on KVM [75] for virtualization. KVM turns the Linux kernel into a hypervisor without need for any additional software. In addition to the typical process operating modes (kernel space, user space) of Linux, KVM adds a *guest mode* for programs running in a virtualized environment. This feature helps for measuring the CPU time used by a virtual process.
As KVM does not perform any emulation to run operating systems on various architectures, we combine it with QEMU[9] to emulate different CPU and device types. With QEMU/KVM, the VM runs as a normal user process and is hence controlled by the Linux scheduler. By default, the scheduler tries to keep a process on the same CPUs, notably to maximize cache efficiency. We run KVM/QEMU with an off-the-shelf Ubuntu 13.11 on both server types (i3 and Xeon).

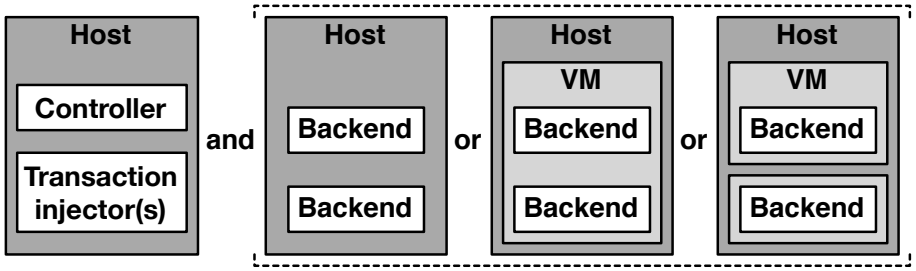

Figure 5.10: Possible setup of SPECjbb (only backends are part of the evaluation).

We want to investigate in our experiments the accuracy and applicability of BitWatts at different scales. Therefore, we first consider the execution of benchmarks on a single host, with an increasing number of concurrently running VMs, to observe the impact of VM scheduling on the host. As a first benchmark, we use PARSEC [74]

[9] http://www.qemu.org, accessed on 17.07.2016

v2.1 for our experiments, as it is multi-threaded and CPU-intensive. PARSEC contains a variety of applications implemented in C. We experiment with all except two (raytrace, ferret) that were not readily supported by our hosts. We use the PARSEC *native* workload as it yields sufficiently long execution times. We allocate 2 threads per VM, thus allowing the execution of 4 concurrent VMs on the Xeon.

Then, to further evaluate BitWatts in a real-world, multi-threaded and distributed environment, we use the SPECjbb2013 benchmark [57]. This benchmark implements a supermarket company that handles distributed warehouses, online purchases, as well as high level management operations (data mining). The benchmark is implemented in Java and consists of *controller* components for managing the application and *backends* that perform the actual work. In our experiments, we focus on evaluating the power consumption of the backends, since they can be scaled arbitrarily in virtualized environments. A run takes approximately 45 minutes; it has varying CPU utilization levels and requires at least 2GB memory per backend to finish properly.

In order to have more than one backend run on our instances of i3, we apply the following parameter changes to `specjbb2013.conf`: we reduce the number of customers and products to $50,000$, increase the step-size, and reduce the maximum and minimum duration for phase 2 of the benchmark.[10]

Since we only have several identical servers of type i3, the SPECjbb experiments are only executed on these machines. We compare different setups, running one or two backends on the host or in a VM (Figure 5.10). The distributed setup consists of a controller host and two virtualized or non-virtualized backend hosts (Figure 5.11). Note that in virtualized scenarios one BitWatts instance runs on the host and one in the VM.

[10] Note that these changes make our runs non-compliant, therefore we do not use the SPEC-specific metrics in this work.

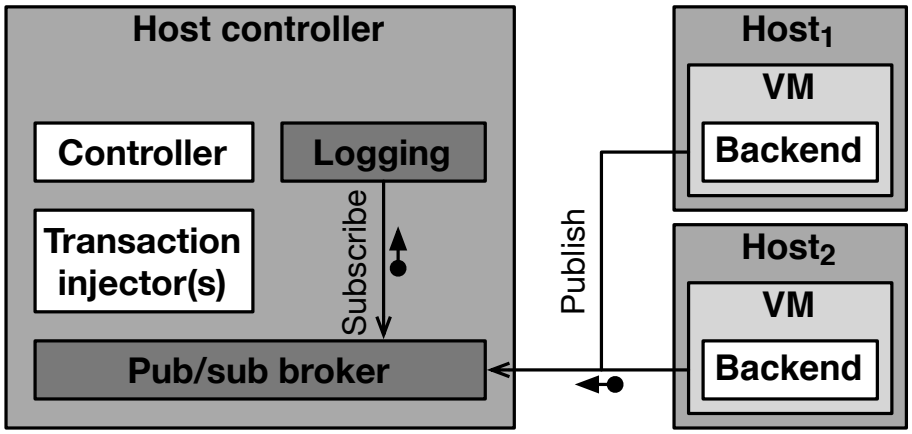

Figure 5.11: Distributed SPECjbb setup.

5.5.2 Scaling the Number of VMs

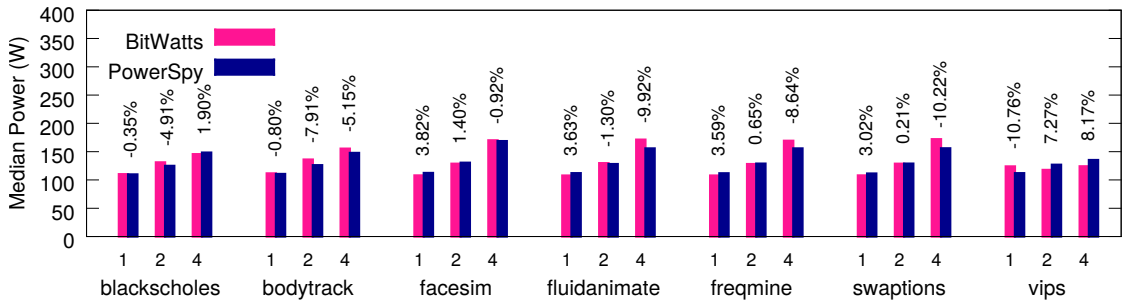

Figure 5.12: Power consumption of the host when scaling PARSEC on multiple VMs.

We already assessed the multi-core CPU power model on the host machine by comparing the BitWatts estimation of PARSEC to the values reported by the PowerSpy. In this section, we first evaluated the virtual CPU power model, described in Section 5.4.2, by comparing the BitWatts estimation of PARSEC running in the VM to the values reported by the PowerSpy on the host. In this experiment, PARSEC is running in a single VM, which has been allocated 2 cores on the host. As the activity of the other active processes is comparably negligible, we compute the BitWatts estimation as the sum of the power estimation in the VM with the idle consumption ($P_{idle}(f)$) of the host machine. Figure 5.12 therefore reports on the median power error observed between BitWatts and PowerSpy. The overall PARSEC experiment resulted in roughly 10,000 power values with a runtime of 1 hour per VM experiment. Note that we did not pin the VM to any specific cores on the host, hence we rely

on the native KVM scheduling. When running a single VM on the host, power estimation within the VM by BitWatts has similar precision to that on the host. This measure assesses that the multi-core CPU power model we propose properly captures the *guest mode* used by KVM to execute the VMs on the host.
Then, given that nowadays VM-based systems tend to be consolidated to minimize the number of active hosts (*e.g.*, [76]), we evaluate the precision of our software-defined power meter when scaling the number of VMs to be executed on the host. For each of the PARSEC benchmarks, we evaluate the median power error when scaling the number of VMs form 1 to 4 on the Xeon processor. As we do not try to measure the side effects of host over-provisioning on power, we do not exceed the number of physical cores available on the host.
The relative error reported in Figure 5.12 spans from less than 1 % (fluidanimate) up to around 10 % (swaptions) with increasing errors if the cores used by the VMs reach the number of physical cores on the host. This reflects results found in literature, but in comparison to existing solutions like VMeter[25] we are not only able to report power per VM if multiple VMs run on a single host, but also per process within each VM. This experiment demonstrates that the virtual CPU power model we introduced in Section 5.4.2 holds in virtual environments, given the simplified architecture of the virtual processor exposed by the hypervisor (see Figure 5.9).

5.5.3 Scaling the Number of Hosts

In this section, we evaluate the power consumption of a real-world application (SPECjbb) using BitWatts. In particular, we further show the possibility of estimating workloads on several nodes such as commonly used in cloud environments. Table 5.1 summarizes the experiments we performed using one or two instances of the SPECjbb backend. The controller runs on a separate host and is not part of our evaluations (see Figure 5.10). We used `taskset` to control the CPU affinity of the multi-threaded backend, which we pin to two physical threads in the execution. As a comparison we also run a non-pinned version of the backend on the host (using all available threads) to see the difference in resource utilization. Two dedicated threads are assigned to each VM.

Name	Description
Host	
`1BE.2t`	1 backend pinned to 2 threads
`2BE.2t`	2 backends, each pinned to 2 threads
`1BE.4t`	1 backend with 4 threads
VM	
`1BE.1VM.2t`	1 backend, 2 threads, 1 VM
`1BE.2VM.2t`	1 backend, 2 threads, 2 VMs
`2BE.1VM.2t`	2 backends, each 2 threads, 1 VM
Distributed	
`1BE.4t`	2 hosts, 1 backend, 4 threads
`1BE.1VM.2t`	2 hosts, 1 backend, 2 threads, 1 VM

Table 5.1: Experiments performed using SPECjbb (`BE`: backend, `VM`: virtual machine, `t`: threads).

The workload characteristics can be seen in Figure 5.13, which plots the power estimation of one backend running on one host. One can clearly observe that the estimation of BitWatts follows the same trend as PowerSpy.

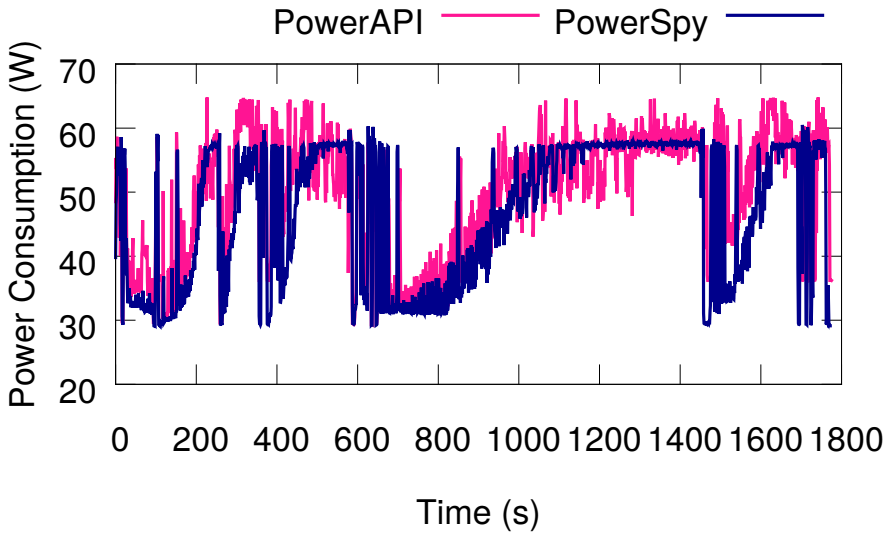

Figure 5.13: Power consumption during the execution of SPECjbb on the *i*3 with 2 threads.

Single node setup. In the literature, applications are usually evaluated in isolated runs. Due to resource sharing, however, process-level estimation becomes more difficult. We further investigate the impact of virtualization as well as interference of concurrently running applications, first on the host and then in virtual machines. In Figure 5.14, we report on the median power consumption of the overall SPECjbb run and the median relative error compared to PowerSpy.
On the host, we run once a backend with all available threads and once pinned to 2 threads to ensure that only some of the CPU cores are used. We can see that the accuracy is not influenced if only a part of the CPU is dedicated to a process. In this experiment, we further show that we can monitor two processes at the same time, when running on the host as well as within the VM. Note that we are monitoring both processes separately and only sum up the process power consumption to compare to PowerSpy. As performance counters interfere when more than one process is running, the isolation of the power consumption for each of the process is harder. This is also reflected in the increasing median error if we monitor more than one process at the same time, *e.g.*, when we run 2 backends on the host or within one or two VMs.

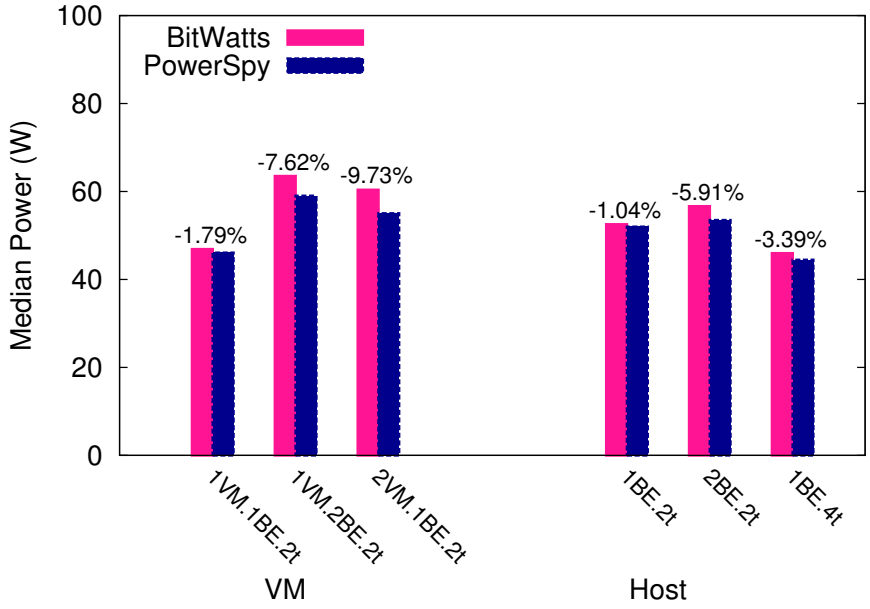

Figure 5.14: Median power consumption for SPECjbb on *i*3 with different resources assigned to a single or multiple VMs on one host.

In the case of the host running only a single backend, we are underestimating the high-load phases (as can be seen in Figure 5.13). In general, however, the estimation

error is below 10 %. BitWatts can therefore also estimate real-world applications with load variations and sub-system scenarios when only parts of the CPU are used. We can further observe that virtualization does not cause power consumption overhead, as can be seen in the single VM run with two backends and the two VMs run with one backend each. KVM is hence very power efficient. We can finally see that the backend can use the available resources more efficiently when it has all threads available (see `1BE.4t` vs. `1BE.2t`) as the highest possible throughput in the workload is reached faster than when the backend has limited resources.

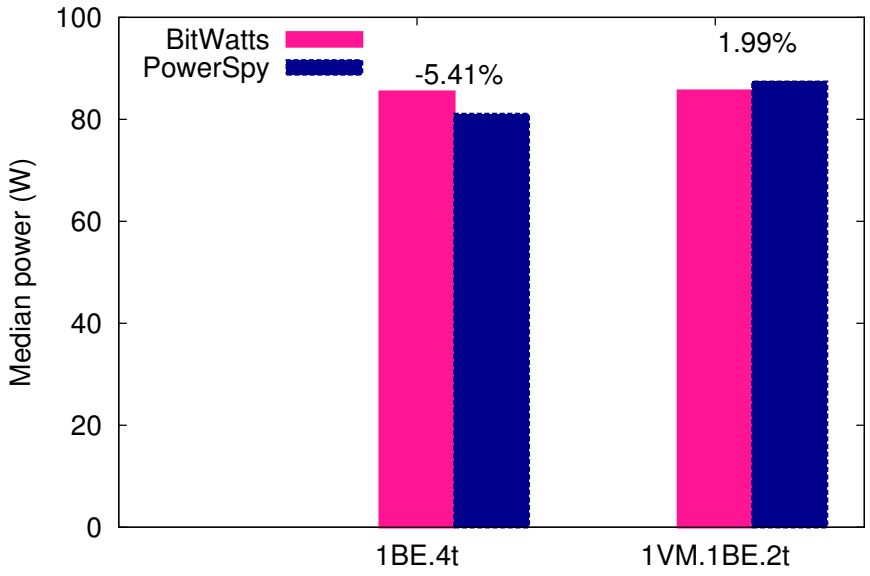

Figure 5.15: Median power consumption for SPECjbb on *i*3 for a distributed setup, virtualized and non-virtualized.

Distributed setup. Placing application components in different VMs allow us to execute across multiple hosts. We therefore extend our experiments to a distributed setup, showing that BitWatts can be applied in realistic data center settings. Experiments were executed on 3 identical servers of type *i*3 as shown in Figure 5.11.
We first run 1 backend on each host, once with 4 available threads, using BitWatts. We also execute 1 backend on 2 hosts, each with a VM and 2 threads. The reporting interval to the broker is 1 s. Based on our observations, the contribution of the network interface to the power consumption is very low and is mainly bound to the CPU activity for sending data. Furthermore, the impact of disk access is not

covered by the SPECjbb benchmark. At the broker, the values are aggregated and forwarded to the logger, which sums the results and writes them to a file.
The results are shown in Figure 5.15. As expected, the absolute power consumption increases with running on two hosts because we have to account for both idle values. The median error, however, does not increase as there are instances of BitWatts on each of the servers and they report the local values to a broker. Furthermore, the single VM experiment shows high accuracy, although underestimating the power consumption. Overall, results are comparable to a single host experiment.

5.6 Summary

In this chapter we presented a middleware, BitWatts, for building software-defined power meters. Such software meters provide an accurate alternative to dedicated hardware systems or embedded power counters by estimating power consumption in the small, *i.e.*, at the level of software processes. With BitWatts we cross the boundaries of virtual environments and provide an estimation of the power consumption of applications running within virtual machines (VMs).
To minimize the estimation error in VMs, BitWatts needs to deliver accurate power estimation for application-agnostic workloads. We therefore developed a CPU power model that considers the complexity of modern processors, including *multi-cores*, *HyperThreading*, *dynamic voltage/frequency scaling*, and *dynamic overclocking* features that impact power consumption. This power model runs in BitWatts without hardware support or system alterations to deliver power estimation with a median error of 2 %. To the best of our knowledge, BitWatts is the first approach to provide such an accurate application-agnostic power model.
Based on this multi-core CPU power model, we proposed a virtual CPU power model that exploits the simplified architecture of virtual processors exposed by the hypervisor to estimate the power consumption of any process running within the VM. The power consumption is forwarded from the host to the VM using an efficient communication channel that connects two instances of BitWatts. It is noteworthy that the proposed architecture can be scaled to multiple levels of virtualization, depending on the complexity of the environment.
BitWatts also supports distributed monitoring setups using publish/subscribe middleware to collect and aggregate power measures reported for several application components, in order to deliver a consolidated view of the consumption of a distributed system.

We evaluated the performance of BitWatts on two processor architectures and in different settings, and we showed that it performs well at different levels of the software stack up to applications.
Since the trend is to run software not only locally, but also in data centers and clouds, additional levels of abstraction have to be considered. Based on an application-agnostic power model that supports the power-aware features of modern processors, we deliver a software-defined power meter to estimate the power consumption of distributed and virtualized setups, which are commonly used in cloud environments. We also demonstrated that our solution is accurate in most cases, even when compared with native information provided by RAPL (see Figure 5.5).
Power consumption is application dependent, hence developers start to take energy into consideration when programming. As a matter of fact, a growing set of tools are created to provide information about the energy efficiency of software [77, 78]. These tools still require direct access to hardware. Since the trend is to run software not only locally, but also in data centers and clouds, we expect BitWatts to represent a valuable contribution for researchers, developers, and engineers. The code is freely available as open source.[11]

[11] http://bitwatts.powerapi.org

6 Impact of Mobile Applications on Battery Life: Power Characterization and Mitigation Measures

In this chapter we study the impact of dynamic power consumption characteristics on battery life and propose a potential solution to extend it. This chapter addresses the layer of the end user applications of the stack presented in Chapter 1.

6.1 Introduction

Our society is increasingly more dependent on mobile devices that run on batteries. These include not just phones, tablets, and portable computers, but also the tiny sensors embedded in all the smart objects that surround us up to larger devices like electrical cars. It is therefore crucial to use energy sparingly in order to extend the lifetime of a battery as much as possible.
At the same time, modern CPUs and the associated co-processors found in modern devices are becoming more and more powerful. They notably include multiple cores with various consumption characteristics, as found for example in the ARM big.LITTLE[1] architecture. ARM big.LITTLE is a heterogeneous computing architecture combining small energy-efficient processors (LITTLE) with others that are faster and more powerful, but also more power-hungry (big). The consumed power consequently varies depending on which core type is used at a given point in time: small or large cores, graphics processing unit (GPU), dynamic voltage and frequency scaling (DVFS), etc.

[1] https://www.arm.com/products/processors/technologies/biglittleprocessing.php, accessed on 17.07.2016

In this chapter, we study how the dynamic characteristics of mobile workloads affect the battery life and propose a solution to alleviate their negative effect. We specifically make three major contributions.

We first conduct an in-depth power characterization of realistic workloads. To that end, we considered a set of classical benchmarks and real-world applications, such as games and Web navigation sessions, running on a recent smartphone. The objective is to find out if power consumption is roughly constant over time, if it follows some regular patterns, or if it exhibits important spikes. This study allows us to get insights into the power consumption of real smartphone applications.

We then evaluate the impact of uneven patterns on the battery life, in order to confirm our intuition that they can negatively affect battery life. We conduct our evaluation using power traces and statistics gathered with real-world applications, as well as on synthetic workloads.

Finally, we propose a solution to mitigate this negative impact and we evaluate its feasibility and benefits. We propose to use a capacitor to reduce the spikes in the power trace and hence protect the battery. We evaluate our approach using an implementation of a state-of-the-art battery model and compare the energy that can be drawn from the battery for the different workloads.

The rest of the chapter is organized as follows: We study the characteristics of mobile workloads in Section 6.2, evaluate the impact of uneven patterns in Section 6.3, and propose our mitigating solution in Section 6.4. We finally discuss our results in Section 6.6 and conclude in Section 6.7.

6.2 Characteristics of Mobile Workloads

In this section we first present the considered workloads and benchmarks. We then define our thesis.

6.2.1 Power Traces

We want to study and characterize the power consumption of real-world mobile workloads. To obtain power traces, we replaced the battery of a recent smartphone, Samsung Galaxy S6, with a physical power meter. At a frequency of 5 kHz, we measured the power consumption of the entire device during the execution of several benchmarks and applications.

We first generated power traces for a number of well-known benchmarks:

- CPU benchmark AnTuTu 5.7.1
- GeekBench v3 (synthetic CPU benchmark)
- Graphics benchmark GFXBench OpenGL (Manhattan workload)
- Unity game engine benchmark (the chase)
- Quadrant benchmark (stresses different components of the phone)

In a second step, we considered applications that are more representative of daily smartphone usage:

- Need for speed
- Website loading and scrolling (typical web surfing behavior)
- PowerPoint presentation with animations
- Angry birds

As an example, Figure 6.1 shows the power consumption of the *powerpoint* workload. We notice that there are high peaks with up to 10 W power consumption, whereas otherwise the power consumption is around 3 W. This is due to the ARM big.LITTLE architecture in the smartphone: during the animation on a specific slide the *LITTLE* core is used, whereas whenever a new slide has to be loaded the *big* core does the work.
We looked at different characteristics of the power traces such as average power, minimum and maximum power, and mean absolute deviation (MAD). Figure 6.2 shows the average, minimum, and maximum power consumption for the different workloads. The minimum and maximum power does indicate the range of power values, but it is not a good indicator for their distribution.
Figure 6.3 shows the MAD, i.e., the average distance of each power value from the average power. We can observe that for example *angrybirds* and *web* have a similar average power consumption, but the MAD is different.

6.2.2 Thesis

We claim that the power trace of a workload is relevant for the battery life. Whenever the MAD is higher, the workload is spread wider around the average power. We posit that a higher MAD leads to more waste of energy: whenever a peak power value needs to be served the discharge current increases and, since the energy that is wasted into

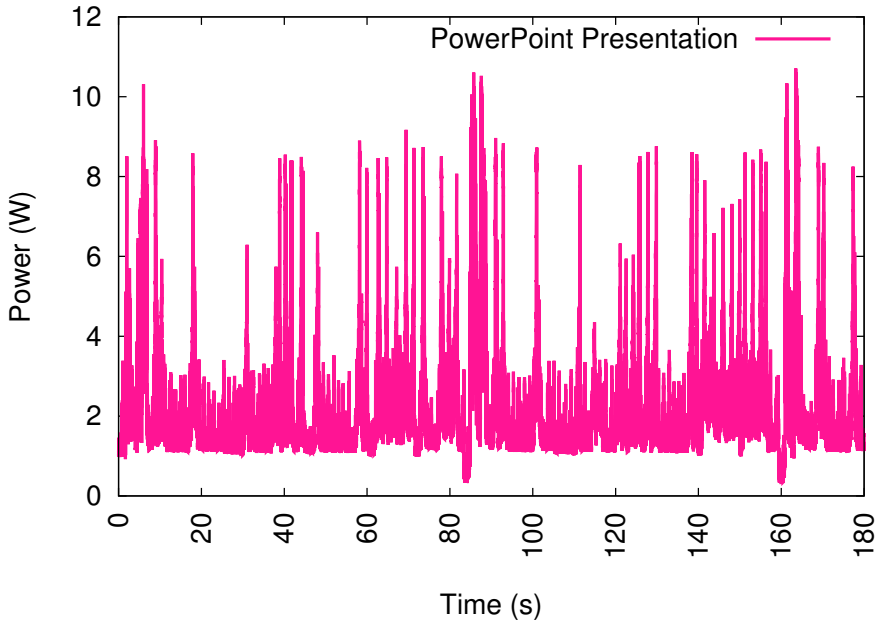

Figure 6.1: Power consumption (W) of the *powerpoint* workload.

heat is proportional to the current, this causes a higher loss of energy. If we are able to reduce peaks in the power trace, we expect the battery life to be extended.

6.3 Impact of Uneven Patterns

We hypothesize that the battery life can be extended by avoiding peaks in the power trace. In the ideal case, we have a completely flat power trace. This corresponds to the average power of a specific power trace. In this subsection we want to evaluate the gain we can obtain with such a flat power curve.

6.3.1 Synthetic Workload

We study a very simplistic workload consisting of phases of high power consumption and phases of idle power. Figure 6.4 shows the corresponding power trace. The pattern is repeated until the battery dies. Figure 6.5 shows the average power and the MAD of the workload comparing to two other benchmarks. One can observe that the *geekbench* benchmark has the highest MAD among the workloads, whereas *unity* has the highest average power.
Figure 6.6 compares the extracted energy for the original power trace and the average power trace, considering one new and one old battery. The old battery differs from

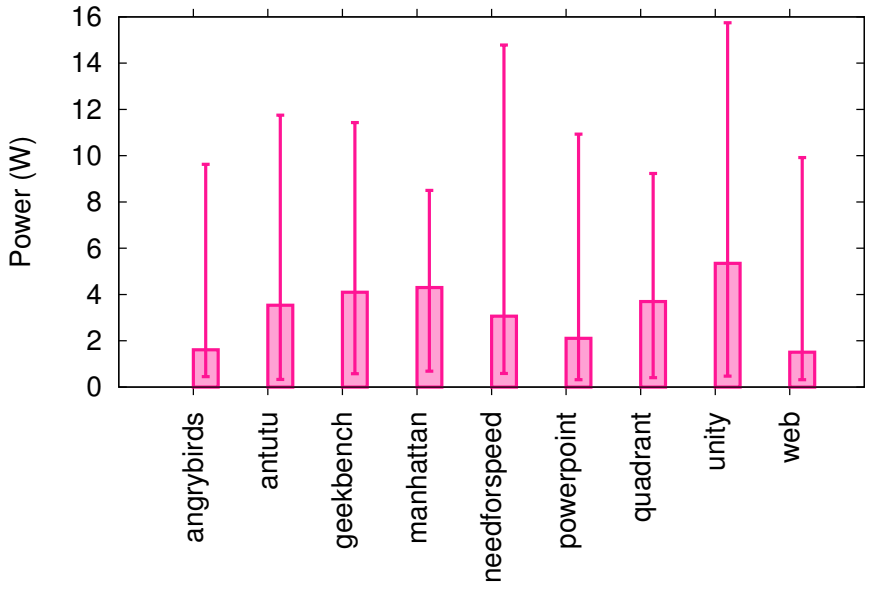

Figure 6.2: Average power (with boxes), maximum power, and minimum power (W).

the new one by having a higher internal resistance. We divide the resulting numbers by the total energy of the battery as specified by the manufacturer's datasheet. This energy can only be extracted from the battery under very specific conditions. We therefore can extract 89% to 92% of the battery energy (or even less for an old battery). The energy gains for *geekbench* and *unity* are less than 1% for the new battery. The simple workload exhibits an improvement of almost 2%. With the old battery, the results are better, because due to the higher resistance, more energy is wasted by power consumption peaks. In the case of the simple workload, the energy gain is almost 5% with the old battery.

6.3.2 Real-World Applications

After the evaluation on the simplistic synthetic workload, we apply our approach to the real-world benchmarks and applications described previously. Figure 6.7 shows the extracted energy for the original power trace and the average power trace, considering one new and one old battery. The difference between the original trace and the average trace is very small for the new battery. When looking at the values we obtained for the old battery with a higher internal resistance, we notice a higher gain, in particular for some workloads such as *geekbench*.

This study of uneven patterns reveals that some benefits—albeit more limited than

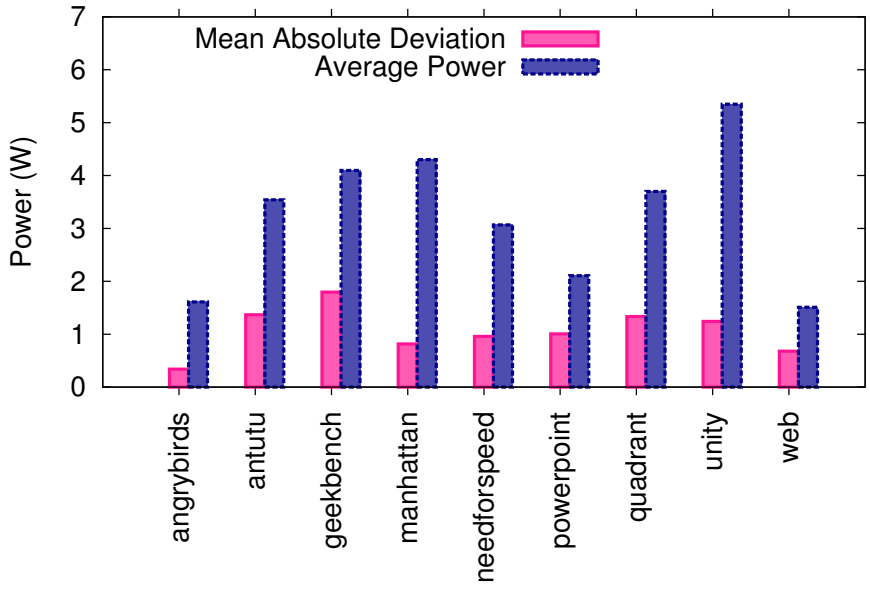

Figure 6.3: Average power and mean absolute deviation (W).

initially anticipated–can be obtained from removing spikes in the power trace. We therefore explore next some possible techniques to smooth out power consumption.

6.4 Using a Capacitor to Reduce Peaks

We now describe in this section a potential solution to the loss of energy caused by peaks in the power trace. Our solution consists in using a capacitor between the battery and the system-on-chip (SoC). Current can also be served from the capacitor during peaks, thus enabling a "battery-friendly" mode. This leads to fewer peaks in the current drawn from the battery and thus more energy can be extracted.

Note that a similar solution was evaluated in a white paper from Texas Instruments [79], where a capacitor operates in parallel with different coin cell batteries to maximize battery capacity utilization. This technique uses the capacitor to offload the power source. During high current periods the capacitor will act as the primary power source. During low current periods the battery is the primary power source and also charges the capacitor.

Figure 6.8 shows our proposed architecture for the passive system. We assume that there is a capacitor between the battery and the SoC to smooth out the power traces. Figure 6.9 shows the architecture of the energy transfer system. In comparison to the previous architecture that applied only a passive filter, this approach adds active

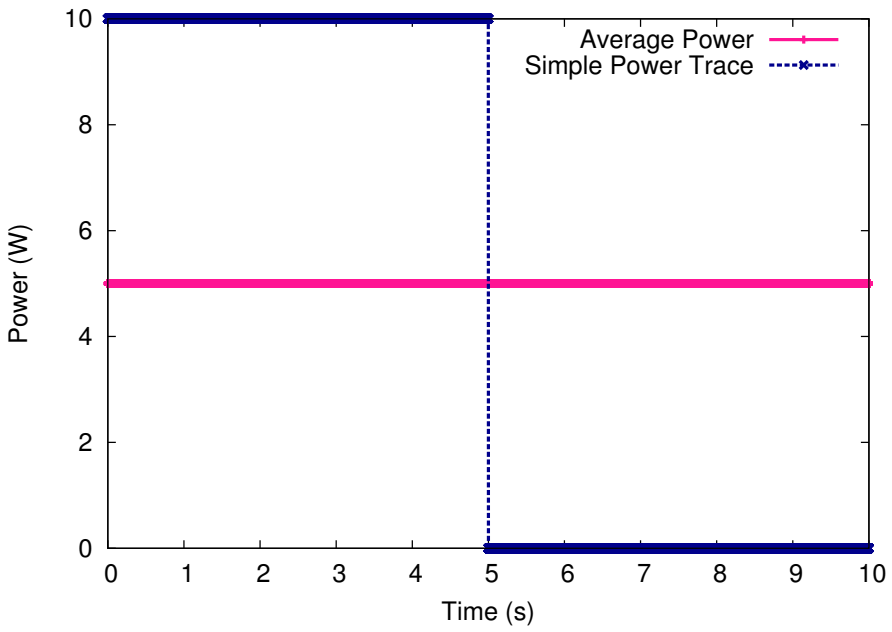

Figure 6.4: Power profile of a simple workload (5000 samples per second).

steering. The system includes a voltage regulator to control the capacitor. Whenever we have to serve a power value that is above average, we discharge the capacitor. This is only possible if there is enough energy stored in the capacitor. In contrast, when the requested power is below average, we charge the capacitor (if it is not yet fully filled). The cut-off could be arbitrary, but we want to get as close as possible to the average power consumption to avoid any peaks. This smarter behavior is expected to deliver better performance than the passive architecture.

6.5 Evaluation

In this section we describe the experimental setup and results for the evaluation of the architecture described in the previous section. We first investigate power traces and identify metrics that can give an indication for the required size and the expected benefits of the capacitor. We then simulate the system with the battery model and a capacitor. We finally complete our evaluation by considering a more complex model.

6.5.1 Setup

The architecture presented in Section 6.4 has been implemented in our simulation environment by applying a low-pass filter with different time constants to the power

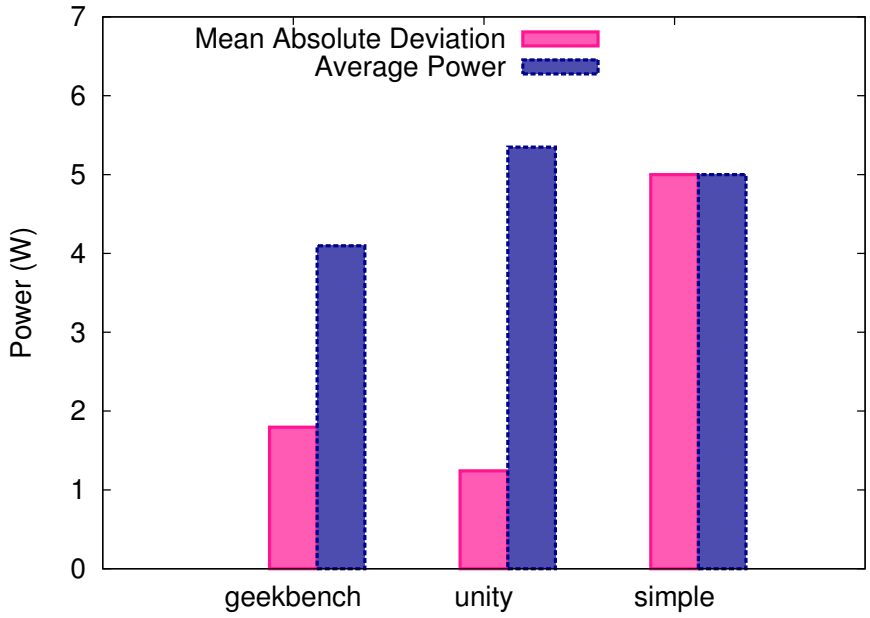

Figure 6.5: Average power and MAD of the simple workload and two benchmarks (*geekbench* and *unity*).

trace. We feed the original power trace and the filtered power traces to the battery model to study the energy that can be extracted from the battery. The power traces are repeated until the battery is discharged. The battery is considered as discharged when a critical capacity value is reached. We evaluated for each workload different sizes of capacitors.

6.5.2 Size of the Capacitor

Figure 6.10 shows a very simple power trace. Intuitively, the size of the capacitor required depends on the size of the *swings* of the peaks in the power curve. More formally, we look at the area of the triangles above and below the average line, which respectively represent the energy above and below average. In Figure 6.10, the energy above average is represented by the darker blue triangles, whereas the energy below average is represented by the lighter orange triangles. Whenever the power trace crosses the average line, we consider this as a *crossing*.

To relate the size of a capacitor and the gain that can be obtained, we place the workloads between the *average energy per crossing* and the *ideal gain*. We want to establish the average energy per crossing as the metric for the size of the capacitor over the effectiveness of the algorithm. The ideal gain is computed by using the results

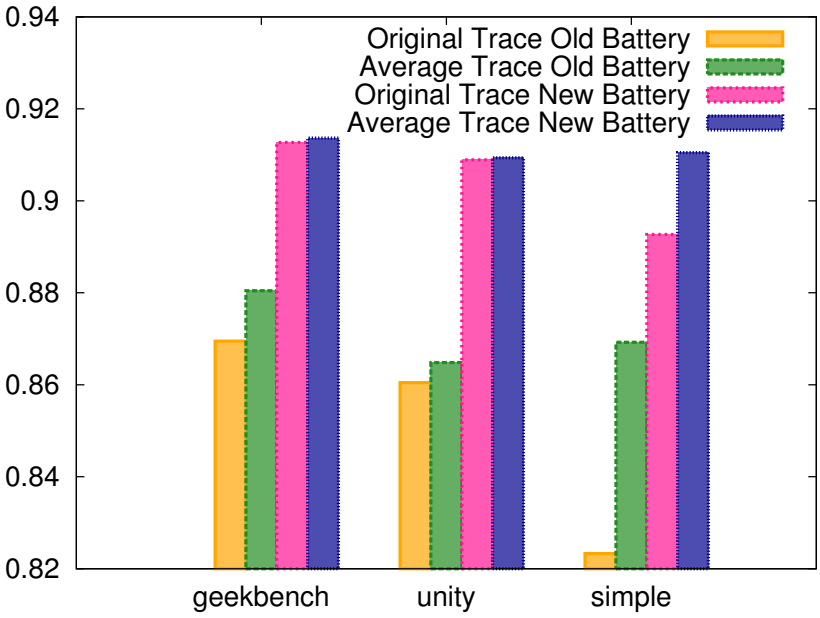

Figure 6.6: Energy gain for the average power trace of the simple workload, and two benchmarks (*geekbench* and *unity*), comparing an old and a new battery. The values are computed by dividing the extracted energy by the datasheet energy of the battery.

of the ideal average power trace compared to the original power trace. Figure 6.11 depicts the results.

If we consider for example the workload *geekbench (gb)*, we observe that it is located in the upper right corner of the graph. This means that we need a rather large capacitor compared to the other workloads (upper part of the graph), and the gain that we can obtain is relatively important compared to the other benchmarks (right part of the graph). The values obtained in terms of gain are unfortunately quite small, even for the ideal case.

The benchmarks that are most interesting are the ones in the lower right corner (i.e., *antutu (ant)*, *quadrant (qu)* and *powerpoint (pp)*). These benchmarks exhibit a good gain compared to the other benchmarks, with a relatively small capacitor.

6.5.3 Passive System

For the passive system we compare the filtered power traces to the original ones. We filtered the power traces with a low-pass filter using different time constants (τ=1 s, 5 s, 10 s) to represent capacitors with different capacitances. A larger time constant represents a capacitor with a larger capacitance. The completely flat average power

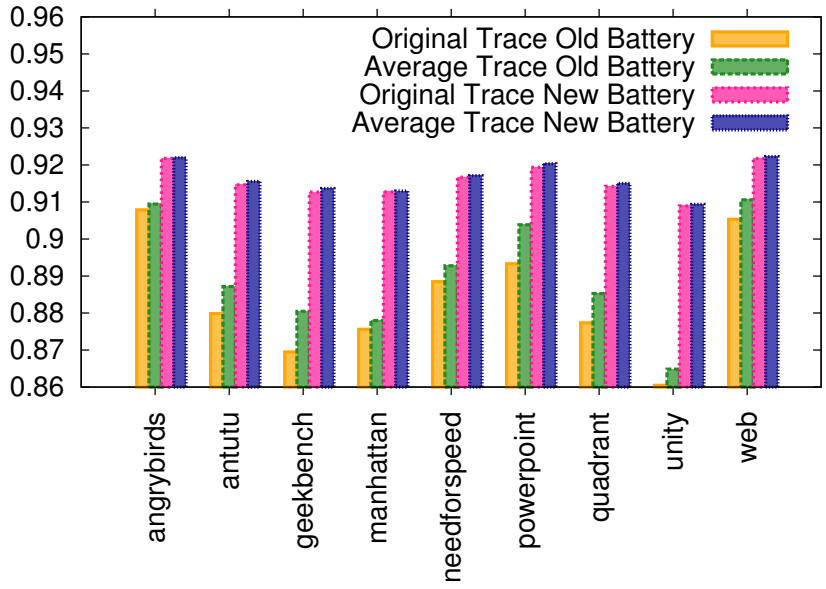

Figure 6.7: Extracted energy of the default power trace and an ideal power trace (average), comparing a new and an old battery. The values are computed by dividing the extracted energy by the datasheet energy of the battery.

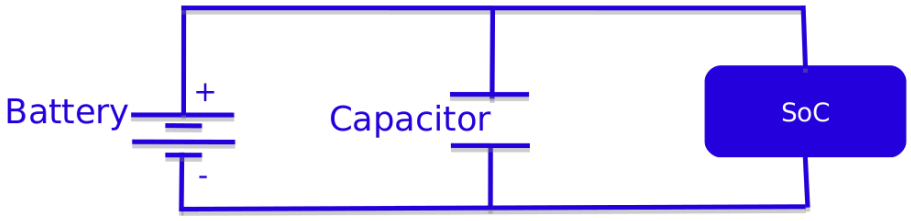

Figure 6.8: Architecture of the passive system.

trace represents an ideal capacitor. The power traces were played into the battery model and repeated until the cut-off capacity of the battery was reached. Figure 6.12 shows the extracted energy in Joule for the original trace, different time constants, and the ideal average power trace for the *geekbench* workload.

The results confirm our expectations. One can extract the most energy with the average trace, and the least with the original trace. The filtered traces are better than the original trace, but not as good as the average power trace. Unfortunately, the absolute energy gain between the original and the ideal approach remains small. Figure 6.13 shows the results for the different benchmarks filtered with a time constant of 10 s, comparing the energy gain over a battery life in Joule. We compare

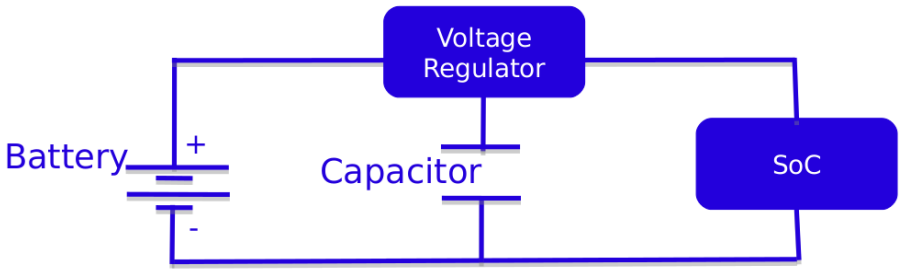

Figure 6.9: Architecture of the energy transfer system.

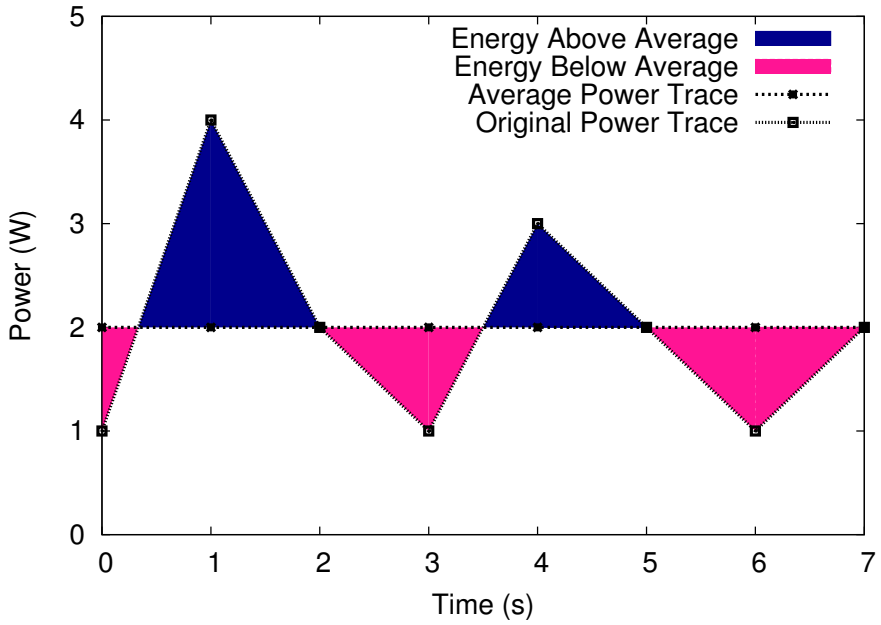

Figure 6.10: Simplified power trace illustrating the concept of crossings.

these values to the MAD. We notice that the trend of the energy gain and the MAD is similar. In most cases, a higher MAD means a higher energy gain with our approach. This confirms our assumption that, when the values in the power traces are spread wider around the average, there is more potential of smoothing the power trace.

6.5.4 Energy Transfer System

In the simulation for the energy transfer system we perform energy accounting for different workloads and different sizes of capacitors. Whenever we need to serve a power value that is above or below average, we respectively discharge or charge the capacitor, if possible. We consider the number of discharges that were possible over the number of potential discharges. In the experiments we consider capacitors

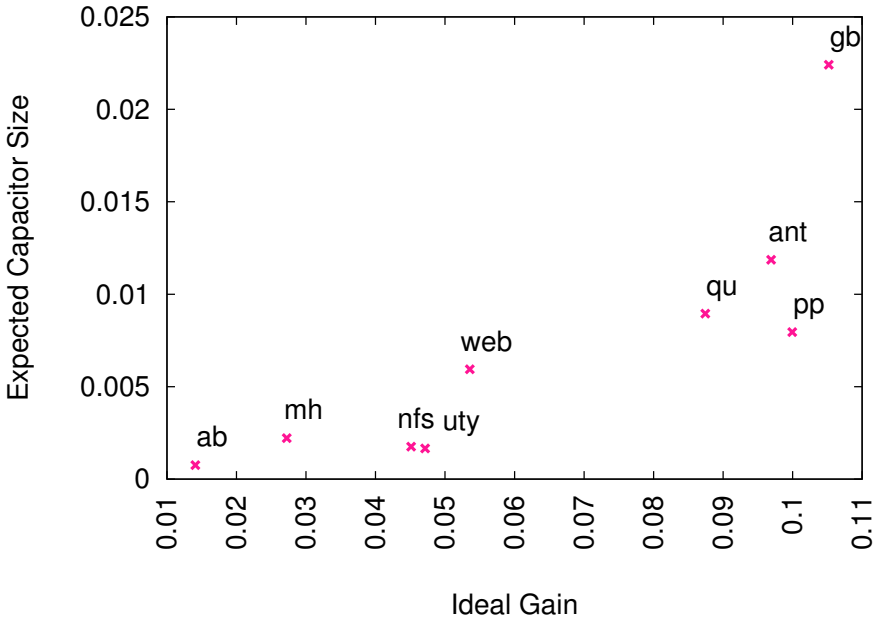

Figure 6.11: Expected capacitor size vs. ideal gain. The expected size of the capacitor is represented by the average energy per crossing (J). The ideal gain is computed by the (extracted energy of the average trace - extracted energy of the original trace) / extracted energy of the original trace (%).

ranging from a capacitance of 0.001 F to 2.0 F. Even though a capacitor of 2.0 F would be physically too large to be used in a smartphone, we include it in the results as a proof of concept.

Figure 6.14 shows the results for the *geekbench* workload. A capacitor of 1.0 F can be discharged in around 70% of the cases. This means, that in 30% of the cases, we would need to discharge it but it is empty. To make our approach more efficient, we would thus require a larger capacitor. This confirms the results from Figure 6.11, where we stated that the *geekbench* workload requires a relatively large capacitor compared to the other workloads.

Figure 6.15 shows the same results for the *angrybirds* workload. A capacitor of 1.0 F can be discharged in more than 90% of the cases. This indicates that a capacitor of 1.0 F is sufficient for this workload. Again, we can confirm the results from Figure 6.11 where we found that *angrybirds* needs a smaller capacitor than *geekbench*, for example.

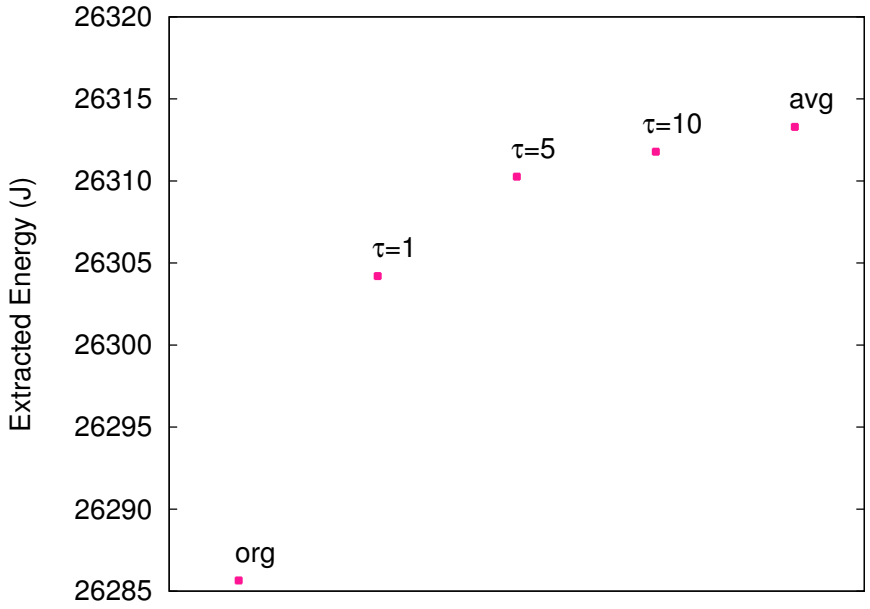

Figure 6.12: Energy gain for *geekbench* comparing the original trace to filtered traces with different time constants and the average power trace.

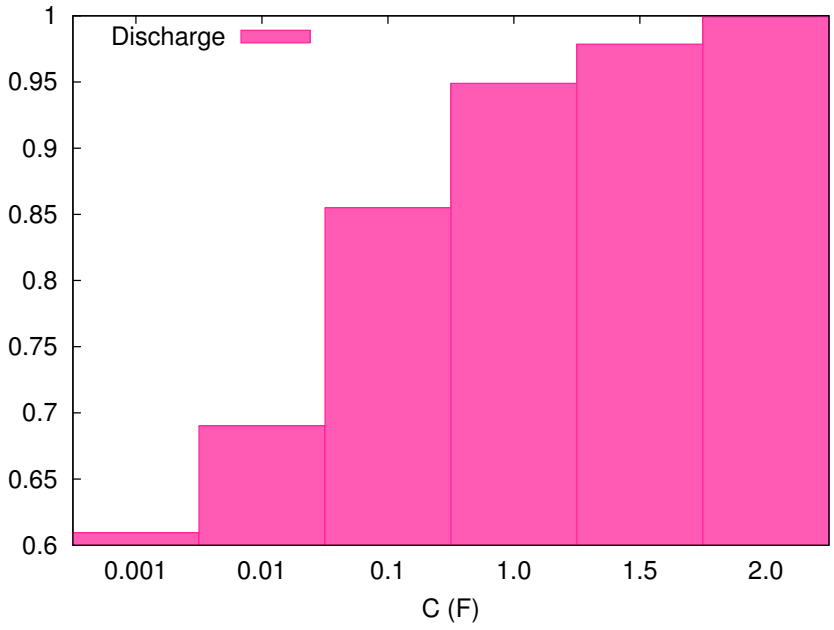

Figure 6.15: Different capacitors for the *angrybirds* workload.

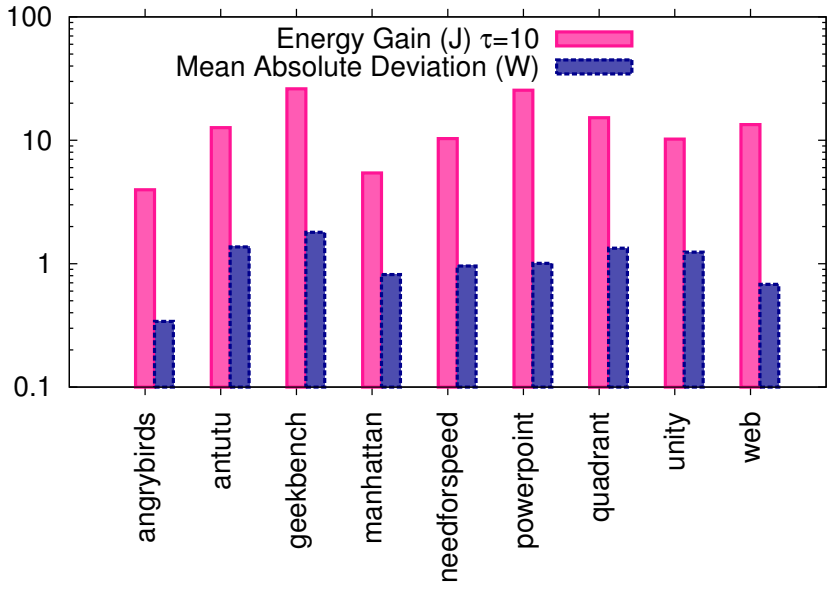

Figure 6.13: Energy gain of the passive system compared to the MAD.

6.6 Discussion and Future Work

The results in the previous section have confirmed our thesis but, at the same time, also indicate that our approach does not provide significant improvement for the battery life. In particular, the improvement for the real-world workloads was very small. One reason is that modern batteries (and their models) are already highly optimized, and thus there is less waste of energy than one could expect. Since we took the power traces from a real smartphone from a recent generation, it likely already incorporates energy-saving optimizations in order to enable a longer battery life. We kept the phone with its default configuration for our measurements, without disabling any functionality.

In the results comparing to the "artificial" old battery with a high resistance, we see better results. We can thus assume that in a different context, the approach would become much more interesting. For example, when using very small batteries with a higher resistance, as for example in the field of IoT [80] and energy harvesting [81], our approach could provide more significant benefits. The battery studied in our model has an internal resistance of 0.01 Ω. For the old battery, we assumed a 0.1 Ω internal resistance. If we consider coin cell batteries, we find internal resistances in the range of tens of Ω, e.g., a typical CR2032 lithium coin battery[2] has a capacity

[2]http://data.energizer.com/PDFs/cr2032.pdf, accessed on 17.07.2016

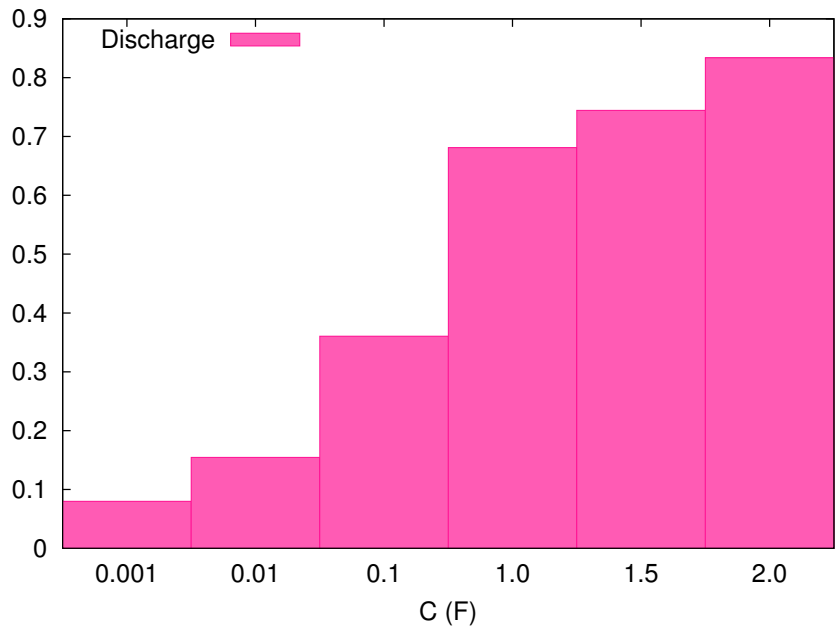

Figure 6.14: Different capacitors for the *geekbench* workload.

of 240 mAh and an internal resistance of 10-40 Ω. Considering the limited gain we obtained, we would therefore need to reconsider the domain of application of our approach.

Furthermore, several factors are not captured by our model, notably battery aging and the impacts of temperature. When a battery has been charged and discharged many times, some of its characteristics change. Electrochemical batteries tend to decrease their charge capacity [82]. In addition, their internal resistance increases, what reduces the amount of power that they can deliver [82] and thermal stability is reduced (more charging energy is converted to heat). In our simplistic model, we only increased the internal resistance of the old battery. A more accurate model would be required to also take into account the loss of capacity of the battery.

The impact of temperature is not included in our model either. When a battery is discharged, the temperature affects the efficiency of a battery [49]. For example, the discharge curve (voltage over state of discharge) at 23° C is different from the discharge curve at 10° C. Operation at different environment temperatures could be relevant for the efficiency of our approach, especially when considering mobile devices that move with users, both indoor and outdoor.

6.7 Summary

In this chapter we have characterized the power traces of real-world applications for mobile devices. We studied in particular the impact of uneven patterns in the power trace on the battery life, and we came to the conclusion that reducing peaks in the power traces can extend it. A solution could consist in using a capacitor between the battery and the system-on-chip. We evaluated our approach with power traces from a recent smartphone. The results show that our thesis is indeed valid, but the energy gain is very small on a high-end mobile device. We believe, however, that our approach could yield greater benefits in a different context, for example when applied to tiny sensors with smaller batteries. Among the lessons learned from this study, we also found out that battery aging can have an important impact on possible battery optimizations and their efficiency.

7 Conclusion and Future Work

The ICT sector, and in particular cloud computing, is still growing and thus, energy consumption is increasing. In this thesis I focused on increasing the energy awareness in heterogeneous data centers and thus contributed to the reduction of energy consumption in the ICT sector. To get an understanding of the energy consumption of an entire system, we need to consider different layers: (1) Data center, (2) Host/virtualization and (3) End user applications. This chapter summarizes the contributions of this thesis and gives future directions on how the outcome of this work could be extended or applied in the real-world.

7.1 Summary of Contributions

In Chapter 2, I defined basic terminology of power and energy and I discussed the impact of different hardware features on the energy efficiency. I showed that Hyper-Threading, Dynamic Voltage and Frequency Scaling, TurboBoost/TurboCORE and the temperature influence the energy consumption of a workload execution. Furthermore, I discussed the related work. I described the different existing models for CPU and VM power consumption. For data centers, I summarized the existing work regarding heterogeneity, idle power attribution and power attribution. In the end I investigated the state-of-the-art of battery modelling.
In Chapter 1 I raised the following research questions:

1. *Is there a relevant change in the power footprint of an application when it runs on heterogeneous hardware, and how can we take advantage of this for workload placement?*

2. *How can we provide transparent, reproducible and predictive cost calculation in heterogeneous data centers based on the energy consumption?*

3. *Which processes, on the host and in virtual machines, are consuming the most power?*

4. *How can we take advantage of dynamic power consumption characteristics to extend the battery life of mobile devices?.*

In Chapter 3, we studied the energy and performance impact of different microbenchmarks (stressing CPU, memory and disk) and a real-world workload running on heterogeneous machines in a data center. We answered the first research question and confirmed that indeed there is a relevant change in the power footprint of an application when it runs on heterogeneous hardware. Furthermore we showed the importance of proper workload placement for the energy consumption in heterogeneous environments.
We applied the knowledge gained from Chapter 3 in Chapter 4 and propose a potential solution. EPAVE is a model for energy-proportional accounting in VM-based environments. It allows transparent, reproducible and predictive cost calculation for users and cloud providers. It is supported by PowerIndex, a profiling and energy estimation framework. PowerIndex is able to profile the energy costs of a VM on a given server architecture and can then estimate its energy costs on a different one. Together, EPAVE and PowerIndex enable energy-aware pricing models in heterogeneous data centers. In Chapter 4 we answered the second research question by providing transparent, reproducible and predictive cost calculation in heterogeneous data centers based on energy consumption.
Chapter 5 presented BitWatts, a middleware for building software-defined power meters. Such software meters provide an accurate alternative to dedicated hardware systems or embedded power counters by estimating power consumption at the level of software processes. We are able to provide a process-level power estimation for processes running within VMs, without modification to hypervisor or operating system. This is an advantage on the related work that either is invasive or considers the VM as a blackbox. With BitWatts we are able to show which processes, on the host and in virtual machines, consume the most power and thus answered the third research question.
Last but not least we looked into battery modelling in Chapter 6. During a 4-months internship at ARM Research UK, I investigated possibilities to extend the battery life for mobile devices. I studied the impact of dynamic characteristics on battery life by collecting and analyzing real power traces. In particular, I focused on uneven patterns and proposed a potential solution to mitigate their effect. I thus answered the fourth research question by showing how we can take advantage of dynamic power consumption characteristics to extend battery life for mobile devices.

7.2 Future Directions

This thesis opens several interesting research perspectives. We discuss here a few of them.
BitWatts (as shown in Chapter 5), EPAVE and PowerIndex (presented in Chapter 4) enhance the energy awareness in data centers. The vision is a completely energy-aware data center. The base for this is provided by the approaches presented in this thesis. To achieve this goal, these approaches need to be enhanced with even more innovative ideas, as shown in the following sections.

7.2.1 Comprehensive Power Estimation

For the time being, the power estimation framework BitWatts presented in Chapter 5 has been evaluated for Intel architectures only. However, the solution we describe does not rely on Intel-specific extensions and can thus be extended for other architectures. To provide a comprehensive power estimation, the framework would need to be extended to support different vendors such as AMD or ARM. Currently, a limitation in a library used by BitWatts prevented us from executing evaluation on AMD architectures.
Another aspect that needs further investigation is the power estimation of disk-intensive workloads. BitWatts currently focusses on CPU- and memory-intensive workloads, since the CPU is the major power consumer in a typical server. For disk-intensive workloads, further studies and more fine-grained models are required, since the state-of-the-art shows that two hard disks, even of the same model and making, have different power consumption patterns [72]. This can be explained by the journaling, the file system, the operating system and the disk fragmentation state. All these factors would need to be considered to provide accurate disk power consumption models. Furthermore, one would need to raise the question whether we want to consider traditional hard disks or SSD disks, that are mostly replacing the traditional hard disks.
Once BitWatts is extended for different architectures and has been complemented with a more complete disk power estimation model, it can truely provide a comprehensive power estimation.

7.2.2 Towards a Completely Energy-Aware Data Center

Energy saving in a data center consists of two major aspects: We can save energy from the idle infrastructure or by reducing the energy of workloads that are executed.

Our approach EPAVE does not account for energy-saving techniques employed by cloud providers, like switching off idle nodes. Furthermore it does not take into account overcommitment. The cost models presented in EPAVE could be extended to allow overall energy efficiency in the data center.

PowerIndex predicts the power consumption of a workload on a different machine. If the power footprint of the two machines has extreme differences, the estimation might not be accurate. However, in combination with EPAVE, we provide upper bounds of dynamic costs and thus misplacement can be avoided. The models within PowerIndex need to be enhanced to cover a wider range of machines with different power properties.

The power estimation framework provided by BitWatts (presented in Chapter 5) is already able to communicate the power estimation values over the network. For a future energy-aware data center, these approaches can be merged and enhanced to provide a global energy view on the entire data center. A combined version of our monitoring infrastructure and cost models would allow a completely energy-aware data center. We would be aware of the energy consumed by each process on each machine at any time. This enables energy-based pricing models; the costumer could be charged in a fair way for the energy consumption he or she caused.

Off course, since with knowledge comes responsability, in such a completely energy-aware data center, security issues are raising. With a global view on the energy consumption of all processes in the data center, one needs to ensure the privacy of such information. The privacy of energy data is a crucial concern, as it can be used for malicious purposes [83]. The authors show how power signatures of computing activities increasingly leak via the power supply. We thus need to consider power and energy traces as confidential data and protect it, as we do for other security-critical information. For the case of the completely energy-aware data center this means that we need, in parallel to the energy-awareness stack presented earlier in this thesis, security measures for each layer. If we consider a global monitoring service for power data in the data center, we need to protect the data while being transported. Furthermore, we need security in place on the individual machines to protect the data when it is stored and to protect privacy when multiple VMs are running on the same hardware. Security measures would also need to be applied for mobile devices and sensors that are storing and sending energy and power values.

7.2.3 Battery Modelling for Current and Future Devices

The insights we gained with battery modelling in Chapter 6 can be further explored for different types of batteries. In our experiments, we considered smartphone batteries. It would be interesting to extend the experiments further towards up-coming domains such as IoT. We realized in our experiments that our approach is more efficient for batteries with a higher internal resistance. In the field of IoT, often smaller batteries with higher internal resistances are used [80] thus this could provide an interesting context for our approach. We also expect that our approach could be exploited in the context of energy harvesting [81].

Also, our simplified model lacks temperature impacts and realistic battery aging behavior. When batteries have been charged and discharged many times, some characteristics as charge capacity, internal resistance and thermal stability change. We consider a very simple aging model based only on the increase of the internal resistance of the battery. Furthermore we are lacking the impact of temperature. The temperature impacts the discharging patterns of a battery. When considering outdoor devices such as sensors or mobile devices, the environment temperature may vary.

To address different contexts and model batteries for a variety of current and future devices, we require a more detailed model. Depending on the characteristics of different batteries, the approach presented in this thesis may achieve better results. Further research could try to implement the mitigation measures presented for different types of batteries in different contexts.

Bibliography

[1] G. Cook, "How Clean is Your Cloud? Report, Greenpeace International, April 2012."

[2] P. Corcoran and A. Andrae, "Emerging Trends in Electricity Consumption for Consumer ICT," *Report from NUI Galway, Ireland.*, vol. 10379, p. 3563, 2013.

[3] G. Cook, "Clicking Clean: A Guide to Building the Green Internet, Greenpeace Internation, May 2015."

[4] T. Heath, B. Diniz, E. V. Carrera, W. Meira Jr, and R. Bianchini, "Energy Conservation in Heterogeneous Server Clusters," in *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pp. 186–195, ACM, 2005.

[5] M. Kurpicz, A.-C. Orgerie, and A. Sobe, "How much does a VM cost? Energy-proportional Accounting in VM-based Environments," in *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pp. 651–658, IEEE, 2016.

[6] M. Kurpicz, A.-C. Orgerie, and A. Sobe, "Energy-proportional Profiling and Accounting in Heterogeneous Virtualized Environments," *Elsevier Journal Sustainable Computing, Informatics and Systems.*, submitted.

[7] M. Colmant, M. Kurpicz, P. Felber, L. Huertas, R. Rouvoy, and A. Sobe, "Process-level Power Estimation in VM-based Systems," in *European Conference on Computer Systems (EuroSYS)*, p. 14, ACM, Apr. 2015.

[8] O. Palomar, S. Rethinagiri, G. Yalcin, R. Titos-Gil, P. Prieto, E. Torrella, O. Unsal, A. Cristal, P. Felber, A. Sobe, Y. Hayduk, M. Kurpicz, C. Fetzer, T. Knauth, M. Schneegass, J. Struckmeier, and D. Milojevic, "Energy Minimization at all Layers of the Data Center: The ParaDIME Project," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 684–689, IEEE, 2016.

[9] M. Zelkowitz, *Advances in Computers: Architectural Advances: 69*. Academic Press, 2011.

[10] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, and R. K. Gupta, "Evaluating the Effectiveness of Model-Based Power Characterization," in *Proc. of USENIX Annual Technical Conference (ATC)*, pp. 12–12, June 2011.

[11] M. Hähnel, B. Döbel, M. Völp, and H. Härtig, "Measuring Energy Consumption for Short Code Paths Using RAPL," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, pp. pp. 13–17, Dec. 2012.

[12] T. Li and L. K. John, "Run-time Modeling and Estimation of Operating System Power Consumption," *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, pp. pp. 160–171, June 2003.

[13] G. Contreras and M. Martonosi, "Power Prediction for Intel XScale® Processors Using Performance Monitoring Unit Events," in *Proc. of IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 221–226, Aug. 2005.

[14] W. L. Bircher and L. K. John, "Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events," in *Proc. of IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS)*, pp. 158–168, Apr. 2007.

[15] M. D. Powell, A. Biswas, J. S. Emer, S. S. Mukherjee, B. R. Sheikh, and S. Yardi, "CAMP: A Technique to Estimate Per-Structure Power at Run-time using a Few Simple Parameters," in *Proc. of IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 289–300, Feb. 2009.

[16] S. Wang, H. Chen, and W. Shi, "SPAN: A Software Power Analyzer for Multicore Computer Systems," *Sustainable Computing: Informatics and Systems*, vol. 1, no. 1, pp. pp. 23–34, 2011.

[17] R. Bertran, Y. Becerra, D. Carrera, V. Beltran, M. Gonzílez, X. Martorell, N. Navarro, J. Torres, and E. Ayguadé, "Energy Accounting for Shared Virtualized Environments Under DVFS Using PMC-based Power Models," *Future Generation Computer Systems*, vol. 28, no. 2, pp. pp. 457–468, 2012.

[18] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual Machine Power Metering and Provisioning," in *ACM Symposium on Cloud Computing (SoCC)*, pp. 39–50, June 2010.

[19] D. Versick, I. Wassmann, and D. Tavangarian, "Power Consumption Estimation of CPU and Peripheral Components in Virtual Machines," *ACM SIGAPP Applied Computing Review*, vol. 13, pp. pp. 17–25, Sept. 2013.

[20] Y. Zhai, X. Zhang, S. Eranian, L. Tang, and J. Mars, "HaPPy: Hyperthread-aware Power Profiling Dynamically," in *Proc. of USENIX Annual Technical Conference (ATC)*, pp. 211–217, June 2014.

[21] K. Shen, A. Shriraman, S. Dwarkadas, X. Zhang, and Z. Chen, "Power Containers: An OS Facility for Fine-grained Power and Energy Management on Multicore Servers," in *Proc. of International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 65–76, Apr. 2013.

[22] A. Verma, P. Ahuja, and A. Neogi, "pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems," in *Proc. of ACM/IFIP/USENIX Middleware Conference*, pp. 243–264, Springer, Dec. 2008.

[23] B. Krishnan, H. Amur, A. Gavrilovska, and K. Schwan, "VM Power Metering: Feasibility and Challenges," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, pp. pp. 56–60, Jan. 2011.

[24] R. Koller, A. Verma, and A. Neogi, "WattApp: An Application Aware Power Meter for Shared Data Centers," in *Proc. of ACM International Conference on Autonomic Computing (ICAC)*, pp. 31–40, June 2010.

[25] A. E. Bohra and V. Chaudhary, "VMeter: Power Modelling for Virtualized Clouds," in *Proc. of IEEE International Symposium on Parallel & Distributed Processing (IPDPSW)*, pp. 1–8, Apr. 2010.

[26] S. Janacek, K. Schroder, G. Schomaker, W. Nebel, M. Ruschen, and G. Pistoor, "Modeling and Approaching a Cost Transparent, Specific Data Center Power Consumption," in *Proc. of International Conference on Energy Aware Computing (ICEAC)*, pp. 1–6, Dec. 2012.

[27] J. Stoess, C. Lang, and F. Bellosa, "Energy Management for Hypervisor-Based Virtual Machines," in *Proc. of USENIX Annual Technical Conference (ATC)*, pp. 1–14, June 2007.

[28] J. Mars, L. Tang, and R. Hundt, "Heterogeneity in 'Homogeneous' Warehouse-Scale Computers: A Performance Opportunity," *IEEE Computer Architecture Letters*, vol. 10, pp. 29–32, July 2011.

[29] C. Delimitrou and C. Kozyrakis, "Paragon: QoS-aware Scheduling for Heterogeneous Datacenters," in *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '13, (New York, NY, USA), pp. 77–88, ACM, 2013.

[30] C. Delimitrou and C. Kozyrakis, "Quality-of-Service-Aware Scheduling in Heterogeneous Data centers with Paragon," *Micro, IEEE*, vol. 34, no. 3, pp. 17–30, 2014.

[31] R. Nathuji, C. Isci, and E. Gorbatov, "Exploiting Platform Heterogeneity for Power Efficient Data Centers," in *Autonomic Computing, 2007. ICAC'07. Fourth International Conference on*, pp. 5–5, IEEE, 2007.

[32] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel, "The Cost of a Cloud: Research Problems in Data Center Networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, pp. 68–73, 2008.

[33] M. Pawlish and A. Varde, "Free Cooling: A Paradigm Shift in Data Centers," in *International Conference on Information and Automation for Sustainability (ICIAFS)*, pp. 347–352, 2010.

[34] A.-C. Orgerie, M. Dias de Assuncão, and L. Lefèvre, "A Survey on Techniques for Improving the Energy Efficiency of Large-scale Distributed Systems," *ACM Computing Surveys*, vol. 46, pp. 47:1–47:31, Mar. 2014.

[35] J. Koomey, "Growth in Data Center Electricity Use 2005 to 2010," *Analytics Press*, 2011.

[36] P. X. Gao, A. R. Curtis, B. Wong, and S. Keshav, "It's Not Easy Being Green," in *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pp. 211–222, ACM, 2012.

[37] D. Snowdon, S. Ruocco, and G. Heiser, "Power Management and Dynamic Voltage Scaling: Myths and Facts," in *Workshop on Power Aware Real-time Computing*, pp. 1–7, 2005.

[38] V. Sivaraman, A. Vishwanath, Z. Zhao, and C. Russell, "Profiling Per-packet and Per-byte Energy Consumption in the NetFPGA Gigabit Router," in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pp. 331–336, April 2011.

[39] E. Pinheiro and R. Bianchini, "Energy Conservation Techniques for Disk Array-based Servers," in *Annual International Conference on Supercomputing*, ICS '04, pp. 68–78, 2004.

[40] A.-C. Orgerie, L. Lefèvre, and J.-P. Gelas, "Demystifying Energy Consumption in Grids and Clouds," in *Work in Progress in Green Computing (WIPGC) Workshop, in conjunction with IEEE IGCC*, pp. 335–342, Aug. 2010.

[41] "How Dirty is Your Data?." Greenpeace report, 2011.

[42] J. Baliga, R. Ayre, K. Hinton, and R. Tucker, "Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport," *Proceedings of the IEEE*, vol. 99, no. 1, pp. 149–167, 2011.

[43] M. Dürr, A. Cruden, S. Gair, and J. McDonald, "Dynamic Model of a Lead Acid Battery for Use in a Domestic Fuel Cell System," *Journal of power Sources*, vol. 161, no. 2, pp. 1400–1411, 2006.

[44] M. Chen, G. Rincón-Mora, *et al.*, "Accurate Electrical Battery Model Capable of Predicting Runtime and IV Performance," *Energy conversion, ieee transactions on*, vol. 21, no. 2, pp. 504–511, 2006.

[45] O. Erdinc, B. Vural, and M. Uzunoglu, "A Dynamic Lithium-Ion Battery Model Considering the Effects of Temperature and Capacity Fading," in *Clean Electrical Power, 2009 International Conference on*, pp. 383–386, IEEE, 2009.

[46] E. Kuhn, C. Forgez, P. Lagonotte, and G. Friedrich, "Modelling Ni-MH Battery Using Cauer and Foster Structures," *Journal of power sources*, vol. 158, no. 2, pp. 1490–1497, 2006.

[47] P. Mauracher and E. Karden, "Dynamic Modelling of Lead/Acid Batteries Using Impedance Spectroscopy for Parameter Identification," *Journal of power sources*, vol. 67, no. 1, pp. 69–84, 1997.

[48] A. Badam, R. Chandra, J. Dutra, A. Ferrese, S. Hodges, P. Hu, J. Meinershagen, T. Moscibroda, B. Priyantha, and E. Skiani, "Software Defined Batteries," in *Proceedings of the 25th Symposium on Operating Systems Principles*, pp. 215–229, ACM, 2015.

[49] L. Gao, S. Liu, R. Dougal, *et al.*, "Dynamic Lithium-Ion Battery Model for System Simulation," *Components and Packaging Technologies, IEEE Transactions on*, vol. 25, no. 3, pp. 495–505, 2002.

[50] O. Tremblay and L.-A. Dessaint, "Experimental Validation of a Battery Dynamic Model for EV Applications," *World Electric Vehicle Journal*, vol. 3, no. 1, pp. 1–10, 2009.

[51] O. Tremblay, L.-A. Dessaint, and A.-I. Dekkiche, "A Generic Battery Model for the Dynamic Simulation of Hybrid Electric Vehicles," in *Vehicle power and propulsion conference, 2007. VPPC 2007. IEEE*, pp. 284–289, IEEE, 2007.

[52] C. M. Shepherd, "Design of Primary and Secondary Cells II. An Equation Describing Battery Discharge," *Journal of the Electrochemical Society*, vol. 112, no. 7, pp. 657–664, 1965.

[53] L. Mashayekhy, M. M. Nejad, D. Grosu, D. Lu, and W. Shi, "Energy-aware Scheduling of MapReduce Jobs," in *Proc. of the 3rd IEEE International Congress on Big Data*, pp. 32–39, 2014.

[54] K. Keahey, M. Tsugawa, A. Matsunaga, and J. A. Fortes, "Sky Computing," *Internet Computing, IEEE*, vol. 13, no. 5, pp. 43–51, 2009.

[55] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano, and I. M. Llorente, "Cloud Brokering Mechanisms for Optimized Placement of Virtual Machines across Multiple Providers," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 358–367, 2012.

[56] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman, *et al.*, "Reservoir - When One Cloud is Not Enough," *IEEE computer*, vol. 44, no. 3, pp. 44–51, 2011.

[57] "SPECjbb2013 Design Document," *Standard Performance Evaluation Corporation (SPEC)*, 2013.

[58] L. A. Barroso and U. Hölzle, "The Case for Energy-proportional Computing," *IEEE Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[59] V. Avelar, D. Azevedo, and A. French, "PUE: a Comprehensive Examination of the Metric." Green Grid white paper, 2013.

[60] U. Institute, "2014 Data Center Industry Survey." `http://journal.uptimeinstitute.com/2014-data-center-industry-survey/`, 2015.

[61] AWS, "Amazon EC2 Pricing." `https://aws.amazon.com/ec2/pricing/?nc2=h_ls`, *accessed in June 2015*.

[62] Q. Liu and D. Zhang, "Dynamic Pricing Competition with Strategic Customers Under Vertical Product Differentiation," *Management Science*, vol. 59, no. 1, pp. 84–101, 2013.

[63] D. Balouek, A. Carpen Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, L. Nussbaum, O. Richard, C. Pérez, F. Quesnel, C. Rohr, and L. Sarzyniec, "Adding Virtualization Capabilities to the Grid'5000 Testbed," in *Cloud Computing and Services Science* (I. Ivanov, M. Sinderen, F. Leymann, and T. Shan, eds.), vol. 367 of *Communications in Computer and Information Science*, pp. 3–20, Springer International Publishing, 2013.

[64] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, *et al.*, "Apache Hadoop Yarn: Yet Another Resource Negotiator," in *ACM Symposium on Cloud Computing (SoCC)*, p. 5, ACM, 2013.

[65] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang, "The HiBench Benchmark Suite: Characterization of the MapReduce-based Data Analysis," in *IEEE Data Engineering Workshops (ICDEW)*, pp. 41–51, IEEE, 2010.

[66] M. Haque, K. Le, I. Goiri, R. Bianchini, and T. Nguyen, "Providing Green SLAs in High Performance Computing Clouds," in *Green Computing Conference (IGCC), 2013 International*, pp. 1–11, June 2013.

[67] A. Amokrane, M. Zhani, Q. Zhang, R. Langar, R. Boutaba, and G. Pujolle, "On Satisfying Green SLAs in Distributed Clouds," in *Network and Service Management (CNSM), 2014 10th International Conference on*, pp. 64–72, Nov 2014.

[68] The Climate Group, "SMART 2020: Enabling the Low Carbon Economy in the Information Age," 2008.

[69] D. Sanderson, *Programming Google App Engine: Build and Run Scalable Web Apps on Google's Infrastructure*. O'Reilly Media, Inc., Dec. 2009.

[70] M. Ben-Yehuda, M. D. Day, Z. Dubitzky, M. Factor, N. Har'El, A. Gordon, A. Liguori, O. Wasserman, and B.-A. Yassour, "The Turtles Project: Design and Implementation of Nested Virtualization," in *Proc. of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 423–436, Oct. 2010.

[71] M. Y. Lim, A. Porterfield, and R. J. Fowler, "SoftPower: Fine-Grain Power Estimations Using Performance Counters," in *Proc. of ACM International Symposium on High Performance Distributed Computing (HPDC)*, pp. 308–311, June 2010.

[72] E. Krevat, J. Tucek, and G. R. Ganger, "Disks Are Like Snowflakes: No Two Are Alike," in *Proc. of USENIX conference on Hot topics in Operating Systems (HotOS)*, pp. 14–14, May 2011.

[73] A. Noureddine, A. Bourdon, R. Rouvoy, and L. Seinturier, "Runtime Monitoring of Software Energy Hotspots," in *Proc. of the IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 160–169, Sept. 2012.

[74] C. Bienia, *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, Jan. 2011.

[75] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "`kvm`: the Linux Virtual Machine Monitor," in *Proc. of the Linux Symposium*, vol. 1, pp. 225–230, June 2007.

[76] T. Knauth and C. Fetzer, "DreamServer: Truly On-Demand Cloud Services," in *Proc. of ACM SIGOPS International Conference on Systems & Storage (SYSTOR)*, pp. 1–11, June 2014.

[77] A. Noureddine, R. Rouvoy, and L. Seinturier, "Unit Testing of Energy Consumption of Software Libraries," in *Proc. of ACM Symposium On Applied Computing*, pp. 1200–1205, Mar. 2014.

[78] C. Sterling, "Energy Consumption Tool in Visual Studio 2013," July 2013.

[79] M. Jensen, "Coin Cells and Peak Current Draw," 2012.

[80] Y.-K. Chen, "Challenges and Opportunities of Internet of Things," in *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, pp. 383–388, IEEE, 2012.

[81] B. Devillers and D. Gündüz, "A General Framework for the Optimization of Energy Harvesting Communication Systems with Battery Imperfections," *Communications and Networks, Journal of*, vol. 14, no. 2, pp. 130–139, 2012.

[82] L. Serrao, S. Onori, A. Sciarretta, Y. Guezennec, and G. Rizzoni, "Optimal Energy Management of Hybrid Electric Vehicles Including Battery Aging," in *American Control Conference (ACC), 2011*, pp. 2125–2130, IEEE, 2011.

[83] S. S. Clark, B. Ransford, and K. Fu, "Potentia est scientia: Security and privacy implications of energy-proportional computing.," in *2012 USENIX Summit on Hot Topics in Security (HotSec '12)*, 2012.