UMEÅ UNIVERSITY

May the Power Be with You

Managing Power-Performance Tradeoffs in Cloud Data Centers

Jakub Krzywda

Doctoral Thesis, August 2019
Department of Computing Science
Umeå University
Sweden

Department of Computing Science
Umeå University
SE-901 87 Umeå, Sweden

http://people.cs.umu.se/jakub
jakub@cs.umu.se

ISBN 978-91-7855-080-7
ISSN 0348-0542
UMINF 19.04

Printed by Cityprint i Norr AB, Umeå, 2019

Abstract

The overall goal of the work presented in this thesis was to find ways of managing power-performance tradeoffs in cloud data centers. To this end, the relationships between the power consumption of data center servers and the performance of applications hosted in data centers are analyzed, models that capture these relationships are developed, and controllers to optimize the use of data center infrastructures are proposed.

The studies were motivated by the massive power consumption of modern data centers, which is a matter of significant financial and environmental concern. Various strategies for improving the power efficiency of data centers have been proposed, including server consolidation, server throttling, and power budgeting. However, no matter what strategy is used to enhance data center power efficiency, substantial reductions in the power consumption of data center servers can easily degrade the performance of hosted applications, causing customer dissatisfaction. It is therefore crucial for data center operators to understand and control power-performance tradeoffs.

The research methods used in this work include experiments on real testbeds, the application of statistical methods to create power-performance models, development of various optimization techniques to improve the power efficiency of servers, and simulations to evaluate the proposed solutions at scale.

This thesis makes multiple contributions. First, it introduces taxonomies for various aspects of data center configuration, events, management actions, and monitored metrics. We discuss the relationships between these elements and support our analysis with results from a set of testbed experiments. We demonstrate limitations on the usefulness of various data center management actions for controlling power consumption, including Dynamic Voltage Frequency Scaling (DVFS) and Running Average Power Limit (RAPL). We also demonstrate similar limitations on common measures for controlling application performance, including variation of operating system scheduling parameters, CPU pinning, and horizontal and vertical scaling. Finally, we propose a set of power budgeting controllers that act at the application, server, and cluster levels to minimize performance degradation while enforcing power limits.

The results and analysis presented in this thesis can be used by data center operators to improve the power efficiency of servers and reduce overall operational costs while minimizing performance degradation. All of the software generated during this work, including controller source code, virtual machine images, scripts, and simulators, has been open-sourced.

Sammanfattning

Det övergripande målet med det arbete som presenterades i denna avhandling är att hitta sätt att hantera prestanda i prestanda i molndatacenter. För detta ändamål analyseras relationerna mellan datacenterservrars strömförbrukning och prestandan hos applikationer som körs i datacenter. Vi utvecklar modeller som fångar dessa relationer och föreslår regulatorer för att optimera användningen av datacenterinfrastruktur.

Studierna motiveras av moderna datacenters massiva strömförbrukning, vilket har betydande ekonomiska och miljömässiga konsekvenser. Tidigare har ett spektrum av olika strategier för förbättring av datacenterets effektivitet stryk föreslagits, inklusive serverkonsolidering, serverstrypning och elbudgetering. Oavsett vilken strategi som används för att förbättra datacenterets effektivitet kan betydande minskningar av energiförbrukningen hos datacenterservrarna med stor sannolikhet försämra prestandan hos applikationer och orsaka kundens missnöje. Det är därför avgörande för datacenteroperatörerna att förstå och reglera prestanda och elförbrukning.

Forskningsmetoderna som används i detta arbete innefattar experiment på riktiga testbäddar, tillämpning av statistiska metoder för att skapa elförbrukningsmodeller, utveckling av optimeringstekniker för att förbättra servrarnas effektivitet och simuleringar för att utvärdera de föreslagna lösningarna i stor skala.

Denna avhandling ger flera bidrag. För det första införs taxonomier för olika aspekter av datacenterkonfiguration, händelser, hanteringsåtgärder och övervakade mätvärden. Vi diskuterar relationerna mellan dessa element och stöder vår analys med resultat från en uppsättning testbäddsexperiment. Vi visar begränsningar på användbarheten av olika datacenterhanteringsåtgärder för att styra strömförbrukningen, inklusive Dynamic Voltage Frequency Scaling (DVFS) och Running Average Power Limit (RAPL). Vi visar också liknande begränsningar på gemensamma åtgärder för att kontrollera applikationsprestanda, inklusive variation av operativsystemets schemaläggningsparametrar, CPU-pinning och horisontell och vertikal skalning. Slutligen föreslår vi en åtgärder för elbudgetering som arbetar på applikations-, server- och klusternivå för att minimera prestandaförlust samtidigt som effektgränser tillämpas.

Resultaten och analysen som presenteras i denna avhandling kan användas av datacenteroperatörer för att förbättra effektivitet och minska de totala driftskostnaderna samtidigt som prestandaförluster minimeras. All mjukvara som genereras i det här arbetet, inklusive regulatorer, virtuella maskiner, skript och simulatorer, har gjorts tillgänglig som open source.

Preface

This thesis contains a brief description of data center infrastructures, a discussion on power-performance tradeoff management strategies with an emphasis on application performance aware power budgeting, and the following papers.

Paper I J. Krzywda, P-O. Östberg, and E. Elmroth. A Sensor-Actuator Model for Data Center Optimization. *Technical Report UMINF 15.20*, Umeå University, Sweden, 2015.
The report is an extended version of a paper that appeared under the same title in *Proceedings of the 2015 IEEE International Conference on Cloud and Autonomic Computing (ICCAC)*, pp. 192–195, 2015.

Paper II J. Krzywda, A. Ali-Eldin, T. E. Carlson, P-O. Östberg, and E. Elmroth. Power-Performance Tradeoffs in Data Center Servers: DVFS, CPU Pinning, Horizontal, and Vertical Scaling. *Future Generation Computer Systems*, Elsevier, Vol. 81, pp. 114–128, 2018.

Paper III J. Krzywda, A. Ali-Eldin, E. Wadbro, P-O. Östberg, and E. Elmroth. ALPACA: Application Performance Aware Server Power Capping. *Proceedings of the 15th IEEE International Conference on Autonomic Computing (ICAC 2018)*, pp. 41–50, 2018.

Paper IV A. V. Papadopoulos, J. Krzywda, E. Elmroth, and M. Maggio. Power-Aware Cloud Brownout: Response Time and Power Consumption Control. *Proceedings of the 2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 2686–2691, 2017.

Paper V J. Krzywda, A. Ali-Eldin, E. Wadbro, P-O. Östberg, and E. Elmroth. Power Shepherd: Application Performance Aware Power Shifting. *Submitted*, 2019.

Paper VI J. Krzywda, V. Meyer, M. G. Xavier, A. Ali-Eldin, P-O. Östberg, C. A. F. De Rose, and E. Elmroth. Modeling and Simulation of QoS-Aware Power Budgeting in Cloud Data Centers. *Submitted*, 2019.

In addition to the papers included in this thesis, the following publications arose from work conducted by the author during his doctoral studies:

Paper VII Ch. Stier, J. Domaschka, A. Koziolek, S. Krach, J. Krzywda, and R. Reussner. Rapid Testing of IaaS Resource Management Algorithms via Cloud Middleware Simulation. *Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering (ICPE 2018)*, pp. 184–191, 2018.

Paper VIII J. Krzywda, W. Tärneberg, P-O. Östberg, M. Kihl, and E. Elmroth. Telco Clouds: Modelling and Simulation. *Proceedings of the 5th International Conference on Cloud Computing and Services Science (CLOSER 2015)*, pp. 597–609, 2015.

Paper IX J. Krzywda, A. Ali-Eldin, P-O. Östberg, A. Rezaie, Z. Papazachos, and R. Hamilton-Bryce. Extended Optimization Model. *CACTOS Project Deliverable D3.3*, 2015.

Paper X P-O. Östberg, H. Groenda, S. Wesner, J. Byrne, D. Nikolopoulos, C. Sheridan, J. Krzywda, A. Ali-Eldin, J. Tordsson, E. Elmroth, C. Stier, K. Krogmann, J. Domaschka, C. Hauser, PJ. Byrne, S. Svorobej, B. McCollum, Z. Papazachos, L. Johannessen, S. Rüth, and D. Paurevic. The CACTOS Vision of Context-Aware Cloud Topology Optimization and Simulation. *Proceedings of the 6th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2014)*, pp. 26–31, 2014.

Financial support for this work was provided in part by the Swedish Research Council (VR) project Cloud Control, the Swedish Government's strategic research project eSSENCE, the European Union's Seventh Framework Programme under grant agreement 610711 (CACTOS), the Swedish Foundation for International Cooperation in Research and Higher Education under grant agreement IB2016-6920, and the European Union's Horizon 2020 research and innovation programme under grant agreement No. 732667 (RECAP).

Acknowledgements

It is now over five years since I have arrived in Umeå. I still clearly remember that dark, rainy, and cold November evening... Luckily, not all the times were so dark—mostly thanks to the people who I have met during my doctoral studies. I have no doubt that without their presence and support I will not be able to finalize this thesis. Therefore, I would like to use this opportunity to express my gratitude.

Towards my supervisors: Erik, for creating an opportunity for me to pursue doctoral studies and guiding me through the valley of the shadow of research; P-O. for his feedback on my research, working on the CACTOS project and teaching the Advanced Distributed Systems course together; and Ahmed for his support, working together on the CACTOS project during the toughest moments, collaboration on research papers, and sharing an office.

César, for establishing an intercontinental collaboration and hosting me during my research stay at Pontifical Catholic University of Rio Grande do Sul in Porto Alegre, Brazil.

Co-authors of my research publications: William, Maria, Trevor, Eddie, Alessandro, Martina, Vinícius, and Miguel, for collaboration, exchanging ideas, and providing feedback.

Current and past members of the Distributed Systems group at Umeå University: Johan, Ewnetu, Monowar, Danlami, Cristian, Thang, Petter, Lars, Chanh, Abel, Tobias, Lennart, Tomas, Simon, Selome, Amardeep, Muyi, Gonzalo, Peter, Mina, Luis, Francisco, Viali, and Daniel, for creating such an interesting, international environment, all our research discussions, lunch talks, afterworks, common trips and all the great moments.

All the colleagues from the Department of Computing Science and HPC2N at Umeå University, especially those with who I had a chance to cross paths at some point over the last five years: Amber, Andrea, Andreas, Andrii, Angelika, Anne-Lie, Bertil, Bo, Birgitte, Carina, Carl Christian, Frank, Helena, Henrik, Jonny, Juan Carlos, Kai-Florian, Lena, Lili, Maitreyee, Mahmoud, Martin, Mats, Mattias, Michael, Michele, Mirko, Monika, Nazanin, Paolo, Pedher, Suna, Sussi, Tarja, Thomas, Thimotheus, Ulrika, Virginia, Xuan, Yvonne, for their help, support, sharing time together during lunches, fikas, and department dinners. Basically, for making my stay at Umeå university easier and more enjoyable.

Colleagues from the CACTOS project: Stefan, Jörg, Craig, James, Henning, Marion, Sergej, Gabriel, Zafeirios, and especially the Runtime Optimisation development team: Christopher, Thanos, Christian, and Darren, for working together, finding all the null pointers exceptions, fighting with CDO technology and afterworks allover the Europe.

Friends from other universities and institutions: Tania, Jonas, Manfred, Tommi, Victor, Gautham, Johan, Albin, Joel, Li, Mulugeta, and Giovanni, for all the interesting discussions and fun times during Cloud Control workshops and other meetings.

Krzysztof Kuchciński for hosting me at Lund University and explaining the secrets of constraint programming.

My teacher from high school Szymon Chucki; professors and advisors from Poznań University of Technology: Andrzej Marciniak, Jacek Kobusiński, Jerzy Brzeziński, Tadusz Morzy, Maciej Piernik; and my supervisor during the internship at INRIA, Peter; for all the knowledge they passed on to me, and for inspiring me to continue my adventure with computer science and research.

My parents for their love and continuous support throughout my life.

Gabriela, for believing in me even when I doubted myself, ultimate support and understanding, care and love.

Contents

Chapter 1

Introduction

The massive power consumption of data centers is concerning because of its financial and environmental impact. Data center operators must spend large amounts of money both to construct power delivery infrastructure capable of handling the constantly growing demand for computation and to pay for the electricity needed to power their equipment. Moreover, many data centers are still supplied with brown energy from polluting sources that generate greenhouse gas emissions and thus exacerbate climate change. Given the importance and contemporary relevance of these issues, there is a clear need to control and minimize the power consumption of data centers.

Because advancements in data center design have greatly improved the power efficiency of cooling and power distribution infrastructures, most of the power used in modern data centers is consumed by servers. Consequently, this thesis focuses on improving the power efficiency of data centers by adjusting configurations at the application and server levels, and by coordinating optimization actions across servers within clusters and across entire facilities.

Several strategies for improving the power efficiency of data centers have been proposed. Since servers achieve their highest power efficiency at high utilization levels, one approach is to consolidate the application workload onto the smallest possible number of physical servers and shut down the empty servers or use them for other processing tasks. However, shutting down servers or using them for other purposes may not be an option, for instance when an application requires all the memory capacity of a cluster, e.g., as in the case of search indexes. Thus, another approach is to throttle servers with low utilization. Throttling is done by reducing a server's power consumption (and thus reducing its performance) to improve its power proportionality. The third approach, power budgeting, enforces limits on the power consumption of data center infrastructures. Power budgeting is commonly used because of the costs of power delivery infrastructures and the billing models of electricity providers, both of which are highly sensitive to the data center's peak power consumption. Therefore, it is in the data center owner's interests to keep peak power consumption low while maximizing

resource utilization. To achieve such high utilization levels, both the data center servers and the power delivery infrastructure are commonly oversubscribed by hosting more applications than they could sustain if all the applications were running at full speed. Consequently, during high workload situations, power limits must be enforced to avoid over-utilization.

To ensure that users of cloud applications are satisfied with the quality of service they receive, a set of non-functional requirements (relating to factors such as availability, security, and performance) must be met. This thesis focuses on the tradeoffs between power, which is discussed above, and application performance. Application performance can be quantified using application-specific metrics such as response time or throughput, which capture the timeliness of service provision. An application's performance depends on several factors, including the characteristics of incoming user requests and the quantity and performance of the computational resources allocated to the application.

Irrespective of the approach used to improve data center power efficiency, substantial reductions in the power consumption of data center servers will reduce the performance of the data center's computational resources (e.g., CPU and memory) and thus affect the performance of hosted applications. However, delivering high and predictable application performance is a key goal of data center operators. Therefore, it is crucial for operators to understand and manage power-performance tradeoffs.

1.1 Research Problem and Objectives

This thesis analyzes the power-performance tradeoffs in data center servers, presents models that capture them, and proposes controllers to optimize data center infrastructures in ways that account for them. The main research problem investigated in this thesis is that of increasing the power efficiency of data center servers while accounting for power-performance tradeoffs.

The overall objective is to find ways of controlling power-performance tradeoffs in data center servers by using appropriate configurations at the application, server, cluster, and data center levels. The main research objectives are:

RO1 To assess the usefulness of various metrics (sensors) and software configuration techniques (actuators) for controlling data center infrastructures.

RO2 To quantify and model the influence of actuators on server power consumption and application performance.

RO3 To develop controllers that optimize the configuration of data center components to achieve a predefined power-performance goal.

Section 5.9 revisits these research objectives and explains how they were achieved.

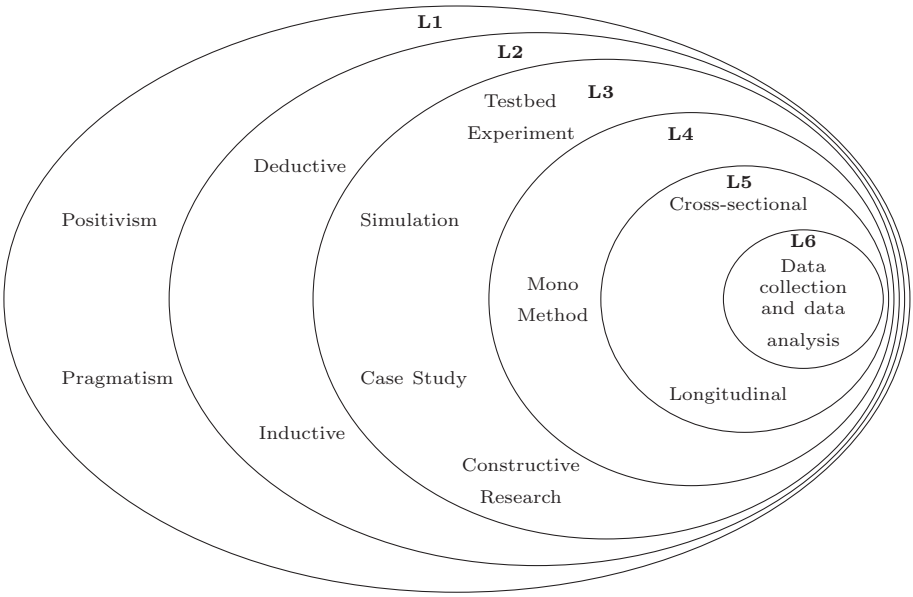

Figure 1.1: Research Onion Diagram.

1.2 Research Methodology

To describe the research methodology adopted in this thesis, we use the *Research Onion* model of Saunders et al. [SLT09]. While this model was primarily developed to structure various aspects of research processes in the field of business and management, it can be also useful in other fields and serves as a framework for reflecting on research.

Figure 1.1 shows the *Research Onion Diagram*. The onion has six layers, namely (starting from the outer layer): **L1**: Philosophical Stances, **L2**: Approaches, **L3**: Strategies, **L4**: Choices, **L5**: Time Horizons, and **L6**: Techniques and Procedures. We have adapted the original diagram by removing items that are irrelevant to the field of computer science in general and this thesis in particular. Below, we describe how each item in the diagram relates to the research presented in this thesis.

Our *philosophical stance* (**L1**) is a mix of positivism and pragmatism. In accordance with *positivism*, we believe that knowledge must be built on empirical evidence. However, our work does not merely involve observing and describing existing phenomena; we also propose improvements to existing solutions. Therefore, like *pragmatists*, we look at the practical consequences of our ideas.

In our research, we use both deductive and inductive *approaches* (**L2**). An *inductive* approach is used, for example, in Paper III, to develop application power-performance models by first collecting data on the power budgets assigned to applications and the applications' performance, and then analyzing those

data to model the relationships between these quantities. On the other hand, we use a *deductive* approach when we try to explain observed phenomena by drawing on resources such as our knowledge of computer architecture. For instance, in Paper II, we first deduce the influence of load balancing strategies on applications' tail response time, then formulate hypotheses, and finally test them by performing experiments and analyzing the results.

We use several different research *strategies* to collect and analyze data (**L3**). Usually, we start our research with a *case study* in which we try to replicate a system we want to understand or improve. We then perform *testbed experiments* that allow us to collect real application performance and power consumption metrics, and model the relationships between them (**RO2**). Next, we apply the *Constructive Research* strategy (CR) [Crn10], also known as *Design Science Research* (DSR) [HKH09]. **RO3** was achieved in this way by creating optimizers and controllers to improve system configuration. These strategies are oriented towards problem-solving rather than focused on accumulation of theoretical knowledge. Moreover, we use *simulation* to evaluate the usability of proposed solutions on scales larger than those achievable on our testbeds.

The studies included in this thesis use only *quantitative* research methods to assess variables such as application performance and server power consumption, and when comparing system configurations. Therefore, our *choice* (**L4**) is a *mono method*, i.e. one that uses either quantitative or qualitative data exclusively.

While time has some impact on our research, it has a less definitive role than that assumed in the concept of the *time horizon* (**L5**) from the research onion, and also a slightly different meaning. On one hand, our experiments and simulations rarely last longer than a few hours. Various changes in the environment occur over the timescale of the experiments, but they are not permanent or even long-lasting. On the other hand, the experiments presented in this thesis were performed on different servers using different technologies. Some of these technologies (e.g. RAPL) have become more widespread recently, which could be considered to make the work implicitly longitudinal. However, none of the direct comparisons we conducted focused on differences related to changes over time. It thus seems that our research has a primarily cross-sectional character.

We use multiple research *techniques and procedures* (**L6**) for data collection and analysis. To obtain data from our testbed experiments, we implement a monitoring framework (for details, see Section 5.8). In Paper II we use a battery of *statistical tests* and quantities to verify that differences between various settings are statistically significant, including descriptive statistics (means and standard deviations), the Welch Two Sample t-test, and one-way analysis of variance for hypothesis testing; Shapiro-Wilk normality tests; and R^2 for regression analyses.

1.3 Research Contributions

This section summarizes the research contributions presented in this thesis and relates them to the research objectives. The specific contributions made by each included paper are described at greater length in Chapter 5.

To address **RO1**, we characterized the spectrum of sensors and actuators available in data centers and proposed taxonomies associating data center events with sensors to monitor them and actuators to respond to them (in Paper I). Further investigations into selected actuators are presented in the later papers.

To address **RO2**, we analyzed the dependence of server power consumption and application performance on various actuators including DVFS, CPU pinning, horizontal and vertical scaling (in Paper II); RAPL, CPU quota/period, and CPU shares (in Paper III); and the percentage of requests served with optional content (in Paper IV). We also quantified the tradeoffs between operational costs and power budget violations (in Paper V), proposed a way to model the relationship between the power budget and application performance, and generated such models for multiple types of cloud applications (in Paper III, Paper IV, and Paper VI).

To address **RO3**, we proposed a set of models to quantify the impact of application performance degradation and electricity consumption on data centers' operational costs. Based on these models, we formulated optimization problems for server power capping (in Paper III), cluster power shifting (in Paper V), and data center power budget adjustment (in Paper VI). We also developed optimization algorithms and controllers to enforce the optimized configurations: ALPACA for prioritizing applications on a single server (in Paper III), Power-Aware Cloud Brownout for adjusting application computational resource demand to the available power budget (in Paper IV), and Power Shepherd for cluster power shifting (in Paper V). Finally, we have implemented an extension to CloudSim for performing simulations to evaluate and compare various power budgeting strategies (in Paper VI).

1.4 Thesis Structure

Chapter 2 provides a high-level description of data center components such as servers, power delivery infrastructure, and types of cloud applications. All these entities form an environment whose operation is monitored using sensors and whose configuration is optimized using actuators. Since the amount of electricity consumed by data centers during their operation is a major concern, Chapter 3 describes various strategies for managing power-performance tradeoffs. For each strategy, we provide an overview of key concepts, challenges, and limitations. In Chapter 4, we focus on one strategy, power budgeting, and explain how to extend it with application performance awareness, creating an environment that supports optimization of power efficiency and data center operation costs. Finally, Chapter 5 summarizes the contributions of the papers forming the

thesis, describes the relationships between them, and reviews the limitations of our approach. The software artifacts developed in connection with the research work are also described.

Chapter 2

Cloud Computing Data Center Infrastructures

This chapter briefly describes the infrastructure of data centers including IT equipment, the power delivery infrastructure, and cooling. We also discuss the power consumption of data centers, its inefficiencies and overheads, and the PUE metric. Next, we review the types of applications that run in data centers. Finally, we present a list of sensors and actuators used to monitor and control power-performance tradeoffs in data centers.

Data centers consume huge amounts of electricity. In 2014, data centers in the United States alone consumed 70 billion kWh of energy, enough to power over six million houses for a year [She+16]. Some estimate that data centers consume about 3% of the world's total electricity supply and have the same carbon footprint as the airline industry (around 2% of total greenhouse gas emissions) [Baw16]. Consequently, it is very important to improve their power efficiency.

Data center infrastructures consist of three main building blocks:

IT equipment, which provides the actual computational and storage services, including computing servers, storage nodes, network components, etc.,

Power distribution infrastructure, which supplies power to the IT equipment, including uninterruptible power supplies (UPS), switchgear, generators, power delivery units (PDUs), batteries, etc.,

Cooling infrastructure, which removes the heat produced by the IT equipment, including chillers, computer room air conditioning units, direct expansion air handler units, pumps, and cooling towers.

Apart from these three main building blocks, data centers need extensive supporting infrastructure including power grids, Internet connections, buildings, roads, and offices for staff. However, these factors are outside the scope of this thesis and therefore not further discussed.

All of the data center infrastructure building blocks listed above consume electricity during operation; their summed consumption constitutes the facility's total energy consumption. Since the ultimate purpose of data center infrastructures is to provide computational and storage services, the energy efficiency of the infrastructure can be defined as the ratio of the energy used for computation to the total energy consumption, which includes (among other things) the overheads of power distribution and cooling [BHR18].

$$\text{Efficiency} = \frac{\text{Computation}}{\text{Total energy}}$$

Energy inefficiencies occur at various levels of the data center infrastructure, ranging from the whole building to individual servers and CPUs. At the level of the whole data center, efficiency is usually quantified in terms of the power usage effectiveness (PUE) [RPC08; TSB06], i.e. the ratio of the power consumed by the facility as a whole to that consumed by the IT equipment.

$$\text{PUE} = \frac{\text{Data center power}}{\text{IT equipment power}}$$

Ideally, the PUE would be 1, indicating that 100% of the power consumed by the data center is used for computation and there are no overheads. It is important to note that PUE measures how much power is used for non-IT equipment. Therefore, any change that only improved the power efficiency of IT equipment would actually *increase* a facility's PUE.

Currently, large state of the art industrial data centers achieve PUE values between 1.11 and 1.06 [Fac19a; Goo19; Fac19b], depending on the measurement boundaries (for example, the precise value depends on whether consumption due to electrical substations and transformers is included in the facility's total power consumption). However, most existing data centers are far behind the state of the art in terms of power efficiency. According to a 2018 Uptime Institute survey covering over 700 data centers of various sizes and geographical locations, the average PUE for current data centers is 1.58 [Asc18].

Improving power efficiency is very important from both financial and environmental perspectives. As shown in Figure 2.1, data center operators have made great efforts over the last decade to achieve current PUE values, having started from an industry average of 2.5 in 2007. For example, by applying DeepMind's machine learning techniques, Google reduced the amount of energy used to cool facilities by up to 40% [EG16]. Over the five-year period from 2011 to 2016, Google increased the amount of computing power it gets per unit energy by a factor of approximately 3.5 [EG16]. Further improvements in PUE are needed to reduce operating costs and/or increase the power available for servers. However, such improvements will require substantial efforts from data center operators, with potentially diminishing returns.

This thesis focuses primarily on computing infrastructure, although power delivery infrastructure is also addressed to an extent. Therefore, the following sections describe the components of these infrastructures in more detail.

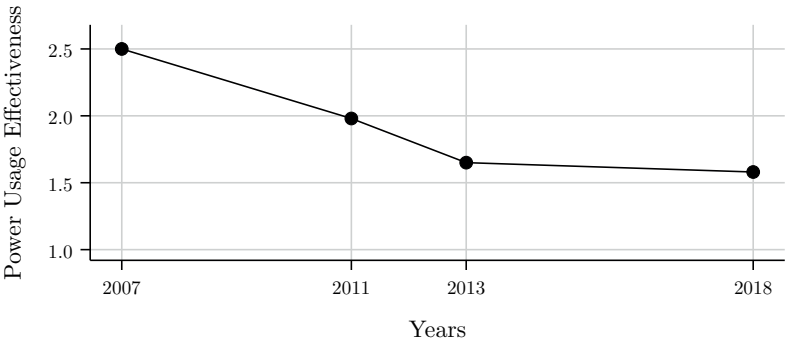

Figure 2.1: Changes in the average PUE for data centers of various sizes and geographical locations over the last decade according to Uptime Institute surveys. Adapted from [Asc18].

2.1 Data Center Servers

The IT equipment used in data centers consists mainly of low-end servers [BDH03], primarily because such servers offer greater cost-efficiency than, for example, high-end shared memory systems. Data center servers typically have multiple CPU cores and hardware support for virtualization, facilitating co-location of multiple applications on a single server.

In the past, data center servers were far from being energy proportional, meaning that their power consumption did not vary linearly with CPU utilization and idle servers consumed significant amounts of power [BH07; Mei+11]. To address this issue, efforts to improve the energy efficiency of data center servers largely focused on two strategies for physical server resource management: server consolidation and server throttling. We discuss these approaches in Chapter 3.

However, the characteristics of data center servers have changed over the last 12 years, as shown in Figure 2.2; modern servers are much closer to achieving the desired energy proportionality. Because cloud data centers usually operate at low utilization levels, it is useful to compare their energy efficiency at utilization levels of 30% (low utilization) and 100% (full utilization). The typical ratio of these efficiencies has improved dramatically over the last decade, from 0.45 in 2009 to 0.80 in 2018 [BHR18].

Several power consumption models for data center servers have been proposed. The simplest are based only on the total CPU utilization and approximate the total power consumption by assuming it varies linearly between the idle and maximum power consumption [BAB12]. More complex models explicitly account for the CPU frequency and voltage, as well as RAM utilization [TWT16]. In Paper II we propose a power model that considers the number of utilized CPU cores and the effects of CPU pinning.

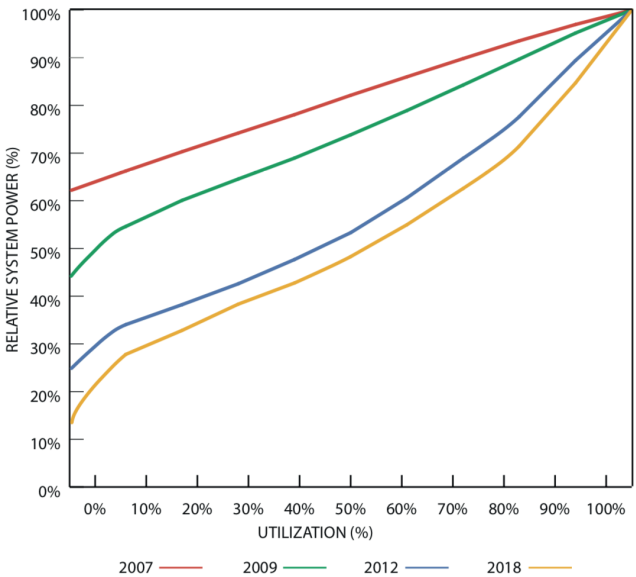

Figure 2.2: Improvement of energy proportionality in Intel servers between 2007 and 2018. Figure from [BHR18].

2.2 Power Delivery

This section describes the power delivery infrastructure that supplies power to data center servers. We also discuss the hierarchy of the power delivery infrastructure and the billing models used by power suppliers, both of which reveal opportunities for optimizing power usage. This thesis does not make any contributions in the field of power delivery infrastructure, but considerations relating to power infrastructure are among the major motivations for the power budgeting approach and the research presented in Papers III–VI. We therefore briefly review this topic for the sake of completeness and as background for these papers.

A data center's power delivery infrastructure is a complex and expensive system whose cost depends on its peak power delivery capacity [FWB07]. The construction of the power delivery infrastructure represents one of the largest capital expenses involved in data center construction; together with the cooling system, the power infrastructure accounts for approximately 60–80% of the total construction cost [BHR18].

The power delivery infrastructures of most commercial data centers are designed to provide high availability. Their design includes redundant components and avoids single points of failure. A typical power delivery infrastructure consists of the following building blocks:

Substation. Because data centers consume a lot of power, they are connected to electrical grids via individual high voltage transmission lines (typically 110 kV and above). However, the power delivery equipment inside data centers operates at a lower voltage (typically below 1 kV). Therefore, the electricity is transformed from high to low voltage in the substations. Sometimes two transformation steps are used, with an intermediate voltage (typically below 50 kV) between the primary utility substation and unit substations distributed within the facility.

Power generator. Longer-term power back-up in case of mains electricity failure is provided by emergency generators, which are usually powered with diesel fuel. However, alternatives running on natural gas and liquid propane also exist [Gen19]. Generators need 10 to 15 seconds to reach their full power output.

Automatic transfer switch (ATS). ATS switches the power source between the main power network and the emergency power generator. The output lines from the ATS are routed to power distribution units via static transfer switches. Power is supplied via two independent routes to increase fault tolerance.

Uninterruptible power supply (UPS). The UPS conditions the power feed and provides a short-term back-up in case of utility outage before the generators achieve their full power. Power conditioning is usually achieved by double conversion between alternating current (AC) and direct current (DC). Power back-up is provided by an energy storage system, which may be electrical (batteries) or mechanical (a flywheel). UPSs are placed on independent routes between the ATS and static transfer switches.

Power distribution unit (PDU). The PDU is a component at the bottom of the power delivery infrastructure hierarchy; it performs the final transformation of the voltage to the value needed by servers and network equipment. Racks of servers are powered directly from the PDUs. PDUs also provide power consumption monitoring capabilities.

Static transfer switch (STS). Each PDU is coupled with an STS that provides redundancy by accepting two independent power routes with connections to two different UPSs. Therefore the failure of a single route or UPS will not interrupt the power supply to the servers connected to the coupled PDU.

More detailed descriptions of power delivery infrastructures are provided in [BHR18; FWB07].

Each component of the power delivery infrastructure has a limited power capacity—the maximum power consumption that it can handle. The power delivery infrastructure of a data center forms a hierarchy with the IT equipment at the bottom and the utility substation at the top. To function properly,

each component of the hierarchy must be able to handle the aggregated power consumption of all components below it. The intuitive way to achieve that would be to take information from the nameplate ratings[1] of IT equipment and propagate it to the top of hierarchy. However, nameplate values are very conservative because they are intended to be taken as upper limits that will not be exceeded. Accordingly, some studies have found that the actual maximum achievable power consumption of a server was below 60% of its nameplate value [FWB07]. For a discussion of misconceptions regarding nameplate ratings, see [Bea+09]. Together with the variable loads of data centers, over-reliance on nameplate data commonly results in underutilization of power delivery infrastructure.

According to a study from 2007 [FWB07], the power consumption of a Google data center is 52–72% of the peak power at the cluster level (5 000 servers). The corresponding ranges for a PDU (800 servers) and a rack (40 servers) are 48–77% and 45–93%, respectively. Consequently, such a data center could potentially host 39% more servers with its existing power delivery infrastructure or could have been built with a less powerful (and thus cheaper) power delivery infrastructure.

To mitigate the problem of underutilized power delivery infrastructure, data center owners have begun accounting for typical power utilization ranges and oversubscribing their power delivery infrastructure. For example, at Facebook data centers, power is oversubscribed at every level of the power delivery hierarchy [Wu+16].

Another important factor that affects the desired power usage of data centers is the electricity provider's billing model. The total cost of electricity usually depends on both the total energy consumed over the billing period (measured in watt hours) and the peak power (measured in watts). The peak power factor can become a substantial part of the final bill, accounting for up to 40% of the total [GSU11]. Thus, when faced with high workloads that may increase the maximum peak power, data center operators must choose between (a) running all servers at full speed but increasing the electricity bill, or (b) throttling the servers to avoid raising the maximum peak power but violating users' service level agreements (SLAs).

2.3 Cloud Workloads

We now shift our attention towards applications running in cloud data centers. The characteristics of these applications and their performance metrics define the performance dimension of the power-performance tradeoffs discussed in this thesis.

[1] The nameplate is a marking indicating a component's compliance with safety standards; it specifies the component's rated input voltage and amperage, which can be used to calculate its theoretical maximum power consumption.

Cloud data centers host a variety of applications with extremely different resource needs, objectives, performance metrics and targets. Some applications handle requests from users that require an immediate response to provide a seamless experience, while others perform background computations with much less strict latency demands. Nevertheless, some common application types can be distinguished to facilitate optimization of their configuration. Moreover, there are multiple ways to deploy applications in clouds, which can influence the actuation techniques that can be used to optimize the data center's configuration.

Below we present a non-exhaustive list of cloud workload types focusing on two aspects that affect the management of power-performance tradeoffs: the application's interactivity level and the deployment model. Additionally, we describe some properties of applications that support these kinds of optimizations.

Interactivity level

One of the most important characteristics of applications hosted in cloud data centers is their interactivity level—how much direct interaction with human users they have. This largely determines their *latency sensitivity*, i.e. the extent to which their quality of service depends on the response time to requests generated by human interactions. Applications can be divided into two classes based on their interactivity levels:

Latency-sensitive services. These are user-facing interactive applications that must satisfy very strict response time requirements to give human users a seamless experience without noticeable delays during interactions. Popular latency-sensitive services deployed in cloud data centers include web search engines (e.g., Google Search), audio and video streaming services (e.g., Spotify and Netflix), and social media platforms (e.g., Facebook).

Batch jobs. These are non-interactive applications that are usually expected to take significant time to generate results. Consequently, their performance is not evaluated in terms of latency. However, they may still have to meet execution time requirements, for example in the form of deadlines. Notable cloud batch processing applications include the Apache big data ecosystem (e.g., Hadoop, Spark, Hive, Flink), machine learning platforms (e.g., Amazon SageMaker, Google Cloud AI, Microsoft Azure AI), and legacy High Performance Computing (HPC) applications.

In our research, we have used both latency-sensitive applications (MediaWiki, Web Search, and Sock Shop) and batch jobs (SysBench) to study various actuators and evaluate proposed optimizations.

Deployment model

Below, we describe popular deployment models and some related technologies.

Bare metal. This is the traditional (pre-cloud era) way of deploying applications without any form of virtualization. Often the whole physical server is used by a single tenant who is both the user of the server and the application owner. Co-located applications share all the resources of the physical server including the operating system. Therefore, co-location is only possible for applications with non-conflicting runtime requirements.

Virtual machines. Virtual machines emulate physical servers and allow co-location of applications with conflicting requirements, e.g. regarding the operating system, on a single physical server. A virtual machine encapsulates an application's code and dependencies as well as the operating system. This deployment model provides a high level of isolation between co-located applications at the cost of potentially higher overheads. Popular virtualization technologies include KVM, Xen, VMware, and Hyper-V.

Containers. Containers are another way of packaging software including the application's code and its dependencies. They are more lightweight than virtual machines because they do not include an operating system—all containers running on a single server share the host operating system. Currently, the most popular container engine is Docker.

Containers are an enabling technology for *microservices*, a software development technique whereby applications are structured as sets of loosely coupled collaborating services. To facilitate deployment of microservices, orchestrators such as Docker Swarm and Kubernetes (K8s) are commonly used to manage interconnections and interactions between individual containers.

This thesis is based on studies that utilized all the deployment models listed above: SysBench as a bare metal deployment, MediaWiki and RUBiS running in virtual machines, and Web Search and Sock Shop deployed in Docker containers; Sock Shop is an application structured as a set of microservices.

Elastic Applications

The usefulness of application-aware power budgeting techniques depends strongly on the ability of the whole system, including both hardware and software, to adapt to dynamic changes in power limits and the associated constraints on computational resource availability and performance. Therefore, applications should ideally be able to handle reductions in CPU frequency or the number of assigned CPU cores while still producing results that are useful to their users.

Applications with optional content. The functionalities of many interactive cloud applications can be divided into two categories: mandatory functions that are crucial for users and optional functions that are nice to have but can be skipped if necessary. If host resource availability is reduced, the response to some user requests can be limited to the mandatory part to provide acceptable latency.

Malleable jobs. In the category of HPC applications, good candidates for supporting power-performance tradeoff management are malleable jobs, i.e. batch workloads that permit dynamic allocation and deallocation of resources at runtime.

Paper IV describes the use of an adapted version of RUBiS (Rice University Bidding System) [OW2+99], in which the percentage of requests served with optional content (recommendations of other products that may be of interest to the user) can be adjusted to match the availability of computational resources.

Having discussed the building blocks of data centers and the characteristics of servers, power delivery infrastructure, workloads, and cloud applications, we now focus on ways of dynamically adjusting a data center's configuration.

2.4 Sensors and Actuators

This section describes various monitored metrics (sensor data) and management actions (actuators) used to control power-performance tradeoffs in data centers.

To understand, model, and control power-performance tradeoffs, we must monitor several metrics relating to applications and the physical status of the data center's infrastructure. The application metrics can be used to quantify performance levels and to determine permissible power-performance tradeoffs based on their inclusion in SLA definitions and the penalties for SLA violations. For example, an SLA may dictate that the average response time for an application must be below a defined threshold and that the data center operator must compensate the application owner if the threshold is exceeded. We discuss this issue further in Section 4.2. The metrics used in this thesis are listed below.

Response time. For many interactive applications, such as Google Search, it is very important that users' requests are served quickly. Here, response time means the time that elapses between a client sending a request and receiving a response. When characterizing an application's performance and aggregating the response times of multiple requests, we distinguish between the average and tail response time. The average response time is typically defined as the arithmetic mean of all successfully processed requests, while the tail response time is defined based on the 95^{th} or 99^{th} percentile of all successfully processed requests.

Throughput. If an application is supposed to be used by many concurrent users, an important parameter is the number of requests the application can serve per unit time. Some of the batch jobs may be even considered throughput critical. Here we define the maximum throughput as the maximum number of application requests served per unit time.

Goodput. If the application is both interactive and accessed by many concurrent users, an essential parameter is the number of requests that can be handled per unit time while maintaining an acceptable response time.

Here we define the maximum goodput as the maximum number of requests served successfully per second while maintaining a response time below an application-specific threshold (e.g., 2 seconds).

Power consumption. Power consumption is measured at two levels in the studies included in this thesis: the whole server and CPU levels. We define the server power consumption as the total power consumption of a physical server measured at a power socket. The CPU power consumption is defined as the power consumption measured by the RAPL module [Dav+10]. In Paper III, we show that the total power consumption differs from the CPU power consumption by a simple constant offset. The power consumption of IT equipment can also be measured at the aggregated levels of rack, PDU, cluster, or whole data center.

In this thesis we evaluate the utility of multiple actuators that can be used to control power-performance tradeoffs. The most important of these actuators are briefly described below. A comprehensive list of data center sensors and actuators is presented in Paper I, while Papers II, III, and IV evaluate the power-performance tradeoffs for a subset of them.

Dynamic voltage frequency scaling (DVFS). DVFS changes the operating frequency and voltage of CPUs to adjust the performance capabilities and power consumption of a physical server [QM11]. The performance capabilities are determined by the CPU frequency, while the power consumption of a CPU depends on both the frequency and the voltage (with voltage having a bigger impact). Each CPU frequency has an appropriate voltage level, and a change of CPU frequency causes a simultaneous change in voltage. For multi-core processors with a single voltage regulator, the appropriate voltage is that required to support the highest CPU frequency of any core at the given time. When we discuss scaling the CPU frequency in this thesis, we implicitly refer to changing both the frequency and the voltage of the CPU.

Running average power limit (RAPL). RAPL [Dav+10] makes it possible to monitor and control the power consumption of CPU sockets and DRAM. RAPL uses a software power model that estimates energy usage based on hardware performance counters and I/O models. Control is exerted by setting a limit on the average power consumption over a user-defined period. The effectiveness of RAPL at controlling server power consumption and its influence on the performance of hosted applications was evaluated by Zhang and Hoffmann [ZH15], who showed that RAPL achieves good stability, high accuracy, and has a low settling time (defined as the time needed for the power consumption to reach and stay close to the desired power budget). However, RAPL can have issues with high maximum overshoot (the actual power consumption can initially exceed the given power limit) and efficiency at low power limits.

CPU pinning. CPU pinning defines the set of CPU cores that each thread (virtual machine) can run on. Pinning can be used to limit the number of CPU cores used by applications in order to reduce power consumption or increase the isolation between co-located applications. By consolidating virtual CPUs on a limited number of physical CPU cores, CPU pinning allows some unused parts of the CPU to enter sleep mode and thereby reduces power consumption. Isolation can be achieved by pinning different applications to different CPU cores (and possibly separate NUMA nodes) to avoid contention for shared resources (e.g., memory bandwidth or last level cache). Podzimek et al. analyzed the effect of CPU pinning on performance interference and energy efficiency for co-located workloads and found that the best setting depends on the CPU load [Pod+15].

Control groups (cgroups). Using cgroups, one can allocate an arbitrary amount of resources (e.g., CPU time, system memory, or network bandwidth) to a collection of processes, including virtual machines. For example, cgroups could be used to define quotas of CPU scheduling time to increase isolation between co-located applications. This makes it possible to throttle applications that do not need resources at a particular point in time or that have lower priority, giving extra CPU time to other applications with greater demands. In our work we mainly use the Linux Completely Fair Scheduler (CFS) parameters: a *CPU quota/period* pair that enforces hard limits on CPU time and *CPU shares* that serve as soft constraints resulting in prioritization relative to other co-located applications.

Vertical scaling. Vertical scaling changes the amount of resources assigned to an application instance. It enables fine-grained adjustment of the number of virtual CPUs and the RAM size of an instance. The main limitation of vertical scaling is the lack of support for several hypervisor actions during virtual machine runtime, e.g., virtual CPU scaling in KVM. Moreover vertical scaling does not permit upscaling above the capacity of the hosting server.

Horizontal scaling. Horizontal scaling changes the number of application instances running concurrently. It is not constrained by the host server's capacity because new instances can be deployed on other physical machines. The main limitation of horizontal scaling, especially when virtual machines are used for application deployment, is that it takes a relatively long time to spawn a new instance because of the need to start an operating system as well as the application itself. Horizontal scaling is also relatively coarse-grained because it requires the deployment or destruction of an entire virtual machine or container to change an application's computing capabilities.

Initial placement. Initial placement is the process of deploying application instances on servers. Initial placement is used for newly admitted services

and for scale outs—deploying new instances of already running services due to horizontal scaling controller decisions. The placement process has two phases: first, a server on which the instance will run, called the *host*, is chosen; second, the instance is deployed on the host. Initial placement techniques for virtual machine are described in detail in Paper I.

Migration. Migration moves application instance from one host to another. This action is useful for data center maintenance actions that require physical machines to be restarted or powered down. Migrations can also be used to optimize the placement of application instances, for example in the server consolidation approach (see Section 3.1). Like initial placement, migration involves two steps: choosing the new host and the actual transfer of the application instance. When applications are deployed inside virtual machines, live virtual migration enables data center operators to transfer virtual machines between servers without interrupting services that the virtual machines provide. Virtual machine migration techniques are discussed in more detail in Paper I. For an in-depth comparison of various approaches to live migration, we recommend [Svä+15].

Application level throttling. Some applications may be implemented in a way that enables them to adapt their execution to the limited amount of available resources, e.g., applications with optional content as discussed in Section 2.3. By adjusting the percentage of requests served with optional content, one can control both the computational resource needs of an application and its performance. In Paper IV, we propose a method for using application level throttling to manage power-performance tradeoffs.

Chapter 3

Power-Performance Tradeoff Management Strategies

Building on the preceding chapter's descriptions of the various components of data center infrastructures, this chapter focuses on ways of improving the power efficiency of the components with the greatest power consumption: the servers. We compare three management strategies: server consolidation, server throttling, and power budgeting. For each strategy, we briefly outline the core concepts, specify the technologies and data center actuators that enable its implementation, review relevant research, and discuss the associated challenges and limitations. Table 3.1 compares the three strategies with respect to their optimization goals and constraints, and the actuators they typically rely on. The sections below discuss each strategy in detail.

Other researchers have presented ways of classifying techniques for improving data center power efficiency. Orgerie et al. [OAL14] consider three levels at which various techniques can be applied, namely the node (e.g., DVFS and sleep state), infrastructure (e.g., workload consolidation and task scheduling), and virtualized environment (e.g., virtual machine migration) levels. This chapter discusses more general approaches that combine techniques implemented at multiple levels to achieve a common goal. Ge et al. [GSW13] distinguish between four general power-saving strategies: energy proportional computing, dynamic provisioning, virtualization, and power capping and shifting. The classification proposed in this thesis is similar, differing mainly in that dynamic provisioning and virtualization are grouped into a single category: server consolidation.

Table 3.1: A comparison of power-performance tradeoff management strategies for cloud computing data centers, showing their typical goals and constraints of optimization as well as commonly used actuators.

Strategy	Optimization goal	Constraints	Actuators
Server consolidation	Minimize power/energy consumption of a cluster	Application performance requirements	VM placement and migration, suspending servers
Server throttling	Minimize power/energy consumption of a server	Application performance requirements	DVFS, CPU pinning
Power budgeting	Minimize operational costs (electricity + QoS)	Maximum power consumption	DVFS, RAPL, power shifting

3.1 Server Consolidation

In principle, server consolidation aims to reduce the number of servers used to host a workload by co-locating applications. This allows some servers to be put into power saving mode or powered down while the rest run at high utilization levels, which is more energy efficient than operating at lower utilization. However, server consolidation has several limitations. First, server consolidation in a data center with dynamic workloads involves migrating virtual machines between servers to reconsolidate the workload onto a minimal subset of servers. Virtual machine migrations are costly—they increase the power consumption of the servers involved in the operation [Hua+11; Jeo+13; Liu+11], negatively impact application performance during the migration time [Liu+11; Voo+09], and increase the utilization of physical resources during the migration [Jeo+13]. Second, powering down physical servers may be impossible because hosted applications may need the resources of a whole cluster to work properly (e.g., memory in the case of Google search [Lo+14]).

The server consolidation approach relies on several data center actuators. Virtual machine migration can be used to fit all the running virtual machines onto a minimal number of servers. The empty servers can then be powered down to eliminate their idle power consumption, or used for other processing tasks. Virtual machine placement algorithms are also needed to determine the location of newly spawned virtual machines and fit them onto already powered up servers. Finally, techniques such as CPU pinning or cgroups may be used to isolate co-located virtual machines and improve their performance.

The goal of server consolidation, to run the whole workload on the smallest possible number of servers, is conceptually straightforward but complex to achieve. The problem of identifying the virtual machine placement that minimizes the number of hosts can be modeled as a (vector) bin packing problem, which is known to be NP-hard [CGJ97]. Therefore many heuristic algorithms

have been proposed that tackle the problem in a greedy way by fitting virtual machines sequentially, starting with the largest, which is intuitively the hardest to fit. Notable algorithms of this type include First Fit Decreasing (FFD) and Best Fit Decreasing (BFD) [Mus+15]. The FFD algorithm was analyzed rigorously and shown to provide solutions that are not worse than 11/9 OPT + 6/9, where OPT is the optimal solution [Dós07]. Beloglazov and Buyya proposed a solution that combines a Modified Best Fit Decreasing (MBFD) algorithm with utilization thresholds to achieve high utilization of data center infrastructure and low performance degradation [BB10]. When dealing with multi-dimensional problems such as those that occur when CPU, memory, storage, and network utilization are all considered, optimization becomes even more complex. In the CACTOS project [Öst+14], we proposed several placement and replacement algorithms, including methods based on constraint programming [Krz+15].

There are still several open challenges in server consolidation that remain to be solved by researchers. Notably, resource fragmentation and stranding phenomena that limit resource utilization must be addressed [VKW14]. Fragmentation occurs when the available resources are spread among several servers but no individual server has enough free resources to host a new service. On the other hand, resources are stranded when there are sufficient free resources of one type on a single server but not enough of some other resource (e.g., when there is ample available memory but no free CPU cores). Jennings and Stadler [JS15] have also shown that predictable performance remains an issue for applications hosted in data centers because of factors such as resource contention and infrastructural heterogeneity, and that virtualization can exacerbate this issue. Mann, on the other hand, suggests further investigation of co-location interference and noisy neighbor effects [Man15]. Finally, most researchers working in this area have focused on optimizing the placement of independent virtual machines, but it should be noted that real applications have many connected and interdependent components.

3.2 Server Throttling

Server throttling involves reducing the power consumption of an individual server at the cost of computational performance. It is used when the server is at a low utilization level, and can therefore be used in conjunction with server consolidation to some extent.

Many techniques (actuators) have been used to throttle the power consumption of data center servers, including Dynamic Frequency Scaling [Gan+09b], DVFS [Kan+14], idle states [MGW09], and CPU pinning [Pod+15]. However, these techniques have indirect effects on both performance and power consumption, making simultaneous control of both parameters a complex task.

The limitations of server throttling are connected to the lack of energy proportionality in data center servers and the difficulty of ensuring adequate performance. General application-agnostic solutions using idle states and DVFS

cannot provide satisfactory power savings with acceptable performance degradation [Kan+14]. Moreover, Lo et al. showed that controlling the performance of latency-critical workloads based on CPU utilization monitoring alone is ineffective, and proposed a feedback-based system that observes the application response time and compares it to a target value [Lo+14]. They also concluded that p-state power management—changing the frequency and voltage of CPU cores during operation—is insufficiently fine-grained.

The most important challenges concerning server throttling that require investigation are those relating to power-performance tradeoffs. There is still a lack of models describing the relationship between server configurations and application performance that can be used to optimize multiple co-located applications. Moreover, Mann points out that it is important to investigate how server throttling techniques interact with virtual machine placement algorithms [Man15].

3.3 Power Budgeting

Power budgeting involves minimizing the capital and operational expenditures of data canter operators by planning and controlling power usage at all stages of a data center's lifetime. This approach exploits knowledge of typical server utilization patterns, the properties of power delivery infrastructures, the billing models of electricity providers, and the ability of modern servers to throttle their power consumption. The main difference between power budgeting and the two previous approaches is that it treats power consumption as a constraint rather than a parameter to minimize.

As noted in Section 2.2, the power delivery infrastructure is one of the most expensive components of a data center, its cost depends on its peak delivery capacity, and it is underutilized most of the time. Based on these observations, a power budgeting approach has been proposed [FWB07; LWW08]. In this approach, data centers are designed with power delivery infrastructures whose capacity is lower than their theoretical maximum consumption. Instead of being prepared to handle a theoretical maximum peak power consumption, the power delivery infrastructure can handle only a *high* data center power consumption, where the consumption limit is chosen to be sufficient in most cases. This substantially reduces the cost of constructing the infrastructure. Moreover, by using power budgeting, the data center operators can control the total peak power consumption, which has a big impact on the electricity bill.

A power budgeting approach typically utilizes two techniques: *power capping* and *power shifting*. Power shifting is used to distribute a total power consumption limit over a group of controlled servers. Then, on each individual server, power capping is used to enforce the local power quota. For example, the power budgeting controller may decide to assign higher power limits to servers hosting high priority services. To achieve that without exceeding the whole cluster's power budget, the power consumption of other servers must

be reduced. Therefore, power capping would be used to throttle the power consumption of severs hosting low-priority services.

Many actuators such as DVFS and CPU pinning can be used for power capping, but they have a major drawback in that they do not limit power consumption directly. More recently, RAPL has been introduced to directly control the power consumption of servers. RAPL ensures that the power consumption limit is enforced, which reduces the complexity of control compared to alternative actuators that indirectly affect both power and performance.

Researchers have examined the problem of power budgeting at various levels of data center infrastructure, including the single server [Coc+11; HM14; Kom+13; LWW08; Liu+16; ZH16], rack [Ran+06; WCF10], cluster [WW11], and whole data center [FF05; Wan+12; Wu+16] levels. Further analysis of recent research concerning power budgeting is provided in related work sections of Papers III–VI.

The main current challenges for the power budgeting approach generally arise from the relationship between the power budget assigned to a server and the performance of hosted applications. The main limitation of power budgeting using RAPL is that power limits are set for whole CPU sockets and not individually for each application. This particularly affects multi-tier services that are spread over multiple servers.

Chapter 4

Application Performance Aware Power Budgeting

Application performance aware power budgeting is the main focus of this thesis. Most of the contributions made in this work, namely those presented in Papers III–VI, relate to this topic. Therefore, this chapter introduces the main concepts and challenges of application performance aware power budgeting, and discusses ways in which it could be achieved.

4.1 Application Performance

Application performance degradation is an inevitable consequence of throttling the power consumption of highly utilized servers. The actual degree of performance degradation varies between applications and also depends on the application performance metric that is used. Moreover, different applications may be valued differently by the data center operator. Therefore, operators may decide to prioritize one group of applications over another when the available power is insufficient to run them all at full speed. In practice, application prioritization can be implemented at one of three granularity levels: server, application, and partial application QoS degradation.

Traditionally, the ranking of applications is determined by the data center operator using static priorities assigned to applications. To facilitate management, servers are also differentiated into priority groups, and the application placement mechanism assigns applications with high priorities to high priority servers. Then, when a power budgeting incident occurs, power is directed towards servers with higher priorities first. Lower priority servers are only allocated power after high priority servers have been allocated sufficient power to ensure that the high priority applications achieve the target performance level.

Alternatively, applications with different priorities can be mixed on individual servers. In such cases, prioritization techniques provided by an operating system scheduler, such as CPU quotas or CPU shares in Linux CFS, can be used to rank hosted applications in order to minimize the performance degradation of the most important ones. As before, applications with the highest priority will be allocated whatever resources they need first, before any resources are assigned to applications with lower priority.

The third approach, which is even more fine-grained, involves degrading the QoS of all hosted applications to some extent, unlike the other two approaches in which the performance of high priority applications will never be degraded before throttling all applications from lower priority groups. This last approach was adopted in the research presented here.

To make power budgeting techniques such as power capping or power shifting aware of application performance, an application performance model is needed to capture the relationship between the power budget and application performance. Such a model could have sufficient predictive capabilities to provide reliable estimates of an application's performance under a modified power budget. Alternatively, one may use a reactive model that identifies actions that can be taken to improve (or degrade) an application's performance. Either type of model could be used to optimize the configuration of data center infrastructures, including applications and servers.

Below, we describe the application-performance-aware management and modeling approaches used in this thesis, as well as other alternatives that have been presented in the literature. Management approaches differ in the inputs they use to control power-performance tradeoffs and the way in which changes are enacted. Additionally, modeling approaches vary in their complexity and the coverage of internal relationships between the hardware and software components affected by power budgeting.

Management Approaches

Management approaches are defined in large part by the inputs that they use for decision making. Several inputs can be used to manage a controlled system, including direct feedback from applications, application-specific models captured offline, application-agnostic feedback from hardware, or a combination of these inputs. Table 4.1 summarizes the differences between management approaches using three basic inputs. Below, we describe each approach in detail.

Application Performance Feedback

An intuitive and straight-forward way of achieving application performance awareness is to expose application performance metrics to a power budgeting controller during the application's runtime. Then, based on the observed metrics, the controller can make on-line adjustments of the application's assigned power consumption limit.

Table 4.1: A comparison of inputs used for decision making, showing their requirements, advantages, and limitations

Input	Requirements	Advantages	Limitations
Application performance feedback *(online)*	Application performance metrics constantly exposed	Robust to new application mixes and workload levels	Not applicable to some application types
Application models *(offline)*	Benchmarking applications beforehand	Applicable to all types of application without any modifications	Prediction quality affected by interference due to co-located applications
Hardware performance counters *(online)*	Hardware and software support for monitoring performance counters at the process level	Non-intrusive monitoring applicable to all application types	Hard to distinguish between ideal and slightly degraded application performance

Control theoretical approaches can be used to model the relationship between the performance metric and the power limit, and to control the performance metric in a predictable manner. The advantage of using feedback is that it is robust to variation in the application's behavior and can sustain a desired state in the presence of external noise such as interference due to co-located applications.

The main limitations of the feedback approach stem from the need for constant exposure of application performance metrics. First, it might be impossible to provide performance metrics for certain application types on a continuous basis. For example, some batch jobs cannot provide reliable progress information until completed. Second, some application owners might be afraid to expose performance metrics during runtime for fear of revealing business-critical information. Third, this approach requires a high level of trust between the infrastructure provider and the application owner. The application owner could potentially abuse their performance feedback to obtain a higher priority by pretending to be adversely affected by power budget reductions even when their application is not actually suffering any performance degradation.

We employ the application performance feedback approach in Paper IV, which describes the use of a cascade controller to manage a power-performance tradeoff for an application with optional content.

Application Models

The second approach uses power-performance models as an input to predict application performance under a given power budget. It requires power-performance

models for hosted applications to be captured offline in advance, during a benchmarking phase, and used for optimization during the application's runtime.

Application models are applicable to all kinds of applications, irrespective of the chosen performance metric. During benchmarking, at the end of each run, the application itself (or a script initiating the application's execution) must produce a measured value of the performance metric, which is then associated with the power budget used during that run. By repeating runs at various power budget levels, one can approximate the relationship between the power budget and application performance. This relationship can then be modeled more accurately by fitting an appropriate function to the measured data points. Models created in this way can be used to predict performance under power budgets that were not examined during the benchmarking phase.

Power-performance models are usually generated separately for each application running in isolation. This reduces the number of configurations that must be evaluated in the benchmarking phase. However, it could reduce the quality of subsequent predictions, which may be significantly affected by interference due to the co-location of applications at runtime.

Moreover, most application models depend on the workload level. This means that during the benchmarking phase, applications should be tested at a range of workload levels corresponding to the range expected at runtime. Therefore, this approach incurs a high modeling overhead.

We employ the application model approach in Paper III, in which we propose ALPACA models. This approach is also applied to power shifting in Paper V, and in large-scale simulations in Paper VI.

Hardware Performance Counters

The third approach involves hardware performance counters—a set of special-purpose registers that can be used to analyze the performance of a whole computer system or selected processes. This approach requires that both the hardware and the operating system support the collection of hardware performance counter data at the appropriate level (process, container, or virtual machine).

An advantage of using hardware performance counters is that their monitoring is non-intrusive and requires no modification of applications. Moreover, it can be applied to any type of application.

The limitations of this approach stem from its hardware and operating system dependence (not all processor models and operating systems support the collection of performance counters at all levels of granularity), as well as the difficulty of translating the counters' output into application performance metrics.

We conducted an initial investigation into the applicability of hardware performance counters, focusing on the Top-Down method [Yas14], while working on a possible extension of Paper III. Unfortunately, the results obtained were

not good enough to warrant further use of this approach to manage power-performance tradeoffs in a runtime environment.

Hybrid Approaches

Some of the approaches described above could potentially be combined to avoid the limitations of single input approaches. For example, one could try to create a hardware performance counter profile of an application offline, during the benchmarking phase, capturing the relationship between the output of hardware performance counters and selected application performance metrics. Later, during the runtime phase, the monitored output of the hardware performance counters could be compared to the profiles to make on-line inferences about current application performance without needing to expose application metrics.

Modeling Approaches

Modeling the performance of applications, and that of computer systems in general, is an important and widely studied problem [Jai91]. Several different approaches have been proposed for analytical modeling [Tay18], including queuing theory [Har13] and Petri nets [Mol82].

The relationship between a server's power budget and the performance of a hosted application is indirect: the power budget affects the performance of computing resources (because it governs variables such as the CPU voltage and frequency), and the performance of computing resources influences that of hosted applications. Therefore, this relationship can be modeled in two ways: by constructing a complex model that links power-resource and resource-application performance models, or by constructing a simple model that tries to directly capture the compound relationship between power and application performance.

Linking Power-Resources and Resources-Performance Relationships

This complex strategy can be regarded as a *white-box approach* in which a cause-effect chain of phenomena occurring within the server is modeled, as shown in Figure 4.1.

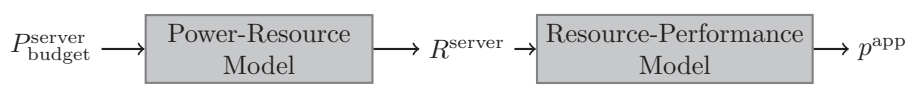

Figure 4.1: The complex approach: modeling the relationship between the server power budget and application performance by linking a power-resource model to a resource-performance model.

The first model has to capture the influence of the server power budget, $P^{\text{server}}_{\text{budget}}$, on the amount and performance of computational resources, R^{server},

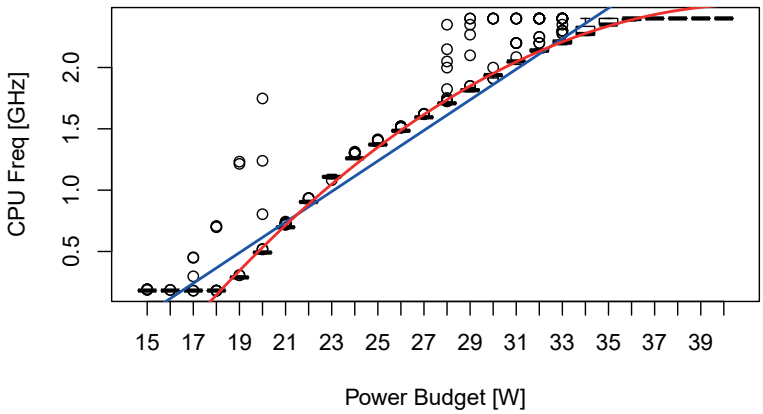

Figure 4.2: Relationship between the RAPL power budget and the CPU frequency, showing both actual measurements and fitted — linear and — quadratic models.

such as CPU, memory, network performance, and so on. While working on Paper VI, we investigated the relationship between the RAPL power budget and the CPU frequency. This relationship was modeled to enable the simulation of hardware performance and its influence on application execution time using existing CloudSim capabilities.

Figure 4.2 shows the relationship between the RAPL power budget given to a package (a CPU socket and the associated DRAM) and the CPU frequency set by the RAPL controller. The operational range of RAPL for a fully utilized server (12 cores at 100% CPU utilization) is between 18 and 36 W, which corresponds to CPU frequencies between 180 MHz and 2.4 GHz. The blue and red lines show, respectively, linear and quadratic regression models of the relationship between the CPU frequency and the power budget over the RAPL operational range. Although the linear model has a high adjusted R^2 of 0.97, indicating a good fit, it systematically over- and under-predicts the CPU frequency at different points along the curve. The quadratic model explains the variation of the CPU frequency better (its adjusted R^2 is 0.99), and its residuals are smaller and more evenly spread along the curve.

To complete the complex approach depicted in Figure 4.1, we require a second model that captures the dependence of application performance, p^{app}, on the amount of computational resources and their performance, R^{server}. The default CloudSim model used in Paper V assumes that the relationship between CPU frequency and application execution time is linear. However, this approach did not yield acceptably precise execution time predictions. Instead, we found that the relationship between CPU frequency and application performance is not

so simple, and may require some kind of black-box modeling to be adequately described.

Direct Power-Performance Relation

When models translating the amount and performance of computational resources into application performance are unavailable for the hosted applications or not readily generated, one can instead capture a model of the direct relationship between the server power budget, $P^{\text{server}}_{\text{budget}}$, and application performance, p^{app}. A method using a simple model of this kind, as depicted in Figure 4.3, can be regarded as a *black-box approach* because it conceals the internal details of the phenomena governing computational resources in power-limited servers.

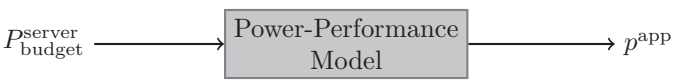

Figure 4.3: The direct approach to modeling the relationship between the server power budget and application performance.

We have modeled the performance of various application types using this approach and used the resulting models to incorporate application performance awareness into power budgeting techniques. Figure 4.4 shows the output of direct power-performance models generated for several applications; for details, see Paper III.

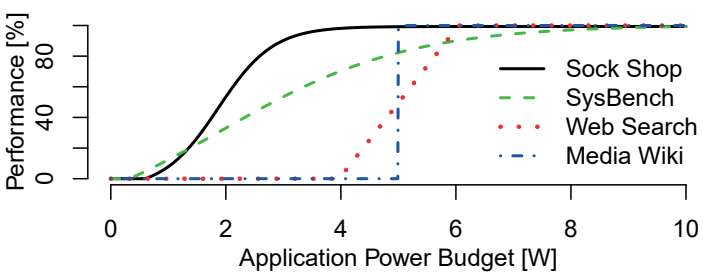

Figure 4.4: Illustrative direct power-performance models. The application power budget is the dynamic component of the RAPL power budget (excluding the power consumption of an idle CPU socket). The application performance is normalized for all applications. From Paper III.

4.2 Application Cost

To evaluate the severity of performance degradation across all applications hosted in a data center, a cost model for quantifying the value of the degradation

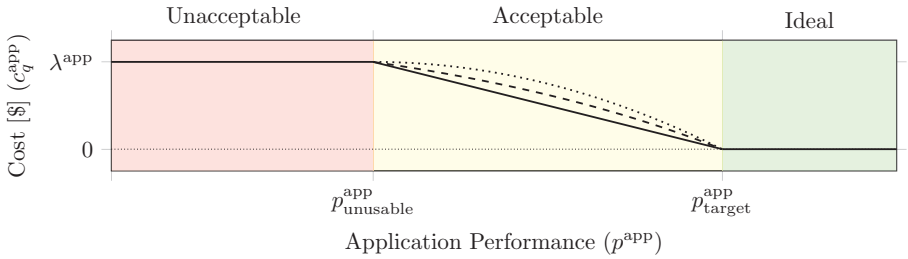

Figure 4.5: The ALPACA application QoS violation model, showing performance thresholds (target and unusable) and costs for various performance degradation functions (— linear, ···· quadratic, and - - exponential).

is needed. Such a model can also be used to find the optimal power budget allocation that minimizes overall costs.

In Paper III, we introduced an application cost model that translates any numerical performance metric into a cost. The model assumes that two thresholds can be defined for every application: a *target performance* level that must be maintained to keep users satisfied with the QoS, and an *unusable performance* level below which the application cannot be considered to be fulfilling its function at all. For example, the end-to-end target response time of a web server should be below 2–3 seconds, giving a delay in page loading that users can tolerate. We assume that delivering performance at the higher level than the target will have no benefit to the service provider. However, when the performance is worse than the target, the application's utility is reduced. At some point, an application's performance may become unacceptable—meaning that its performance is so heavily degraded that its results are unusable. In the case of a web server, its performance becomes unacceptable, for example, when processing a request takes so long that the response arrives after the user has already decided to abandon the service.

Our approach assumes that the infrastructure provider is responsible for the performance degradation of hosted applications. Therefore, when the application performance is below the target, a QoS violation penalty is imposed. To the best of our knowledge, such direct compensation systems are not used in the industry, but it is still important for infrastructure providers to be seen as reliable and credible by their users and such models are relevant to be used inside data center organizations regardless if the models are exposed to their customers. Our model can be used to capture the cost of reduced reliability by representing it in terms of direct financial compensation. It should be noted that our work focuses on the technical aspects of the solution; business and liability models are outside the scope of this thesis.

Figure 4.5 shows the relationship between application performance p^{app} and application cost. Each application has a defined target performance constraint $p^{\text{app}}_{\text{target}}$ and a threshold at which the application becomes unusable $p^{\text{app}}_{\text{unusable}}$.

A violation of a performance constraint results in a penalty. The actual QoS violation cost depends on the level of performance degradation $p^{\text{app}}_{\text{degr}}$, and the application's importance is specified by the maximum penalty λ^{app}.

The QoS violation cost of an application with a performance degradation $p^{\text{app}}_{\text{degr}}$ is then defined as follows:

$$c^{\text{app}}_{\text{q}} = \begin{cases} 0, & \text{if } p^{\text{app}} \geq p^{\text{app}}_{\text{target}}, \\ \lambda^{\text{app}}, & \text{if } p^{\text{app}} \leq p^{\text{app}}_{\text{unusable}}, \\ \lambda^{\text{app}} p^{\text{app}}_{\text{degr}}, & \text{otherwise.} \end{cases}$$

No costs are incurred when the application's performance is better than or equal to the target threshold (corresponding to the ideal area in Figure 4.5). When the application's performance is below the unusable threshold, the maximum cost is imposed (corresponding to the unacceptable area in the figure). For application performance levels between the two thresholds (the acceptable area), the penalty value depends on the chosen performance degradation function.

To capture a wide spectrum of applications, our model supports flexible definition of the performance degradation function. The impact of performance degradation will not necessarily be linearly proportional to the change in application performance. For example, the consequences of the response time increasing from 3 to 4 seconds might be more severe than that of a slow down from 9 to 10 seconds. Therefore, the relationship between an application's performance degradation and its actual performance p^{app} can be defined in various ways, e.g.:

$$p^{\text{app}}_{\text{degr}} = \begin{cases} \frac{p^{\text{app}}_{\text{target}} - p^{\text{app}}}{p^{\text{app}}_{\text{unusable}} - p^{\text{app}}_{\text{target}}}, & \text{linear,} \\ \left(\frac{p^{\text{app}}_{\text{target}} - p^{\text{app}}}{p^{\text{app}}_{\text{unusable}} - p^{\text{app}}_{\text{target}}} \right)^2, & \text{quadratic,} \\ A \ \exp\left(\frac{p^{\text{app}}_{\text{target}} - p^{\text{app}}}{p^{\text{app}}_{\text{unusable}} - p^{\text{app}}_{\text{target}}} \right) - A, & \text{exponential,} \end{cases}$$

where $A = \frac{1}{\exp(1)-1}$.

For all the functions, performance degradation increases from 0 (indicating performance at or above the target threshold) to 1 (indicating performance at or below the unusable threshold).

4.3 Electricity Cost

Electricity cost is one of the main operating costs of data centers. Data center operators have to pay not only for the electricity consumed over the billing period (usually a month), but also for the readiness of the utility provider to supply the declared maximum power that can be consumed by the facility. The exact formula for calculating the electricity bill value varies among countries and even among utility providers. Here, we propose a general electricity cost model that can support a variety of situations.

The proposed complete electricity cost model depends on two main variables:

Total energy consumption, i.e. the aggregated power consumption over the billing period:

$$E = \int P^{\mathrm{dc}} dt.$$

Peak power consumption, i.e. the maximum power consumption over the billing period:

$$PP = \max P^{\mathrm{dc}}.$$

To calculate the actual electricity cost, each of these variables is multiplied by a corresponding unit cost parameter to determine its contribution to the total electricity cost. Finally, the contributions due to total energy and peak power consumption are summed:

$$c_{\mathrm{E}} = \epsilon E + \pi PP,$$

where ϵ is the unit cost per kWh of energy consumed and π is the unit cost per W of peak power consumption.

The complete electricity cost model is used in Paper VI.

Simplified Electricity Cost Model

For optimization problems that do not account for the time dimension and thus make momentary decisions, the electricity cost model can be simplified to an equation that depends on a single variable (the momentary power consumption) and a single unit cost parameter. The simplified electricity cost model is:

$$c_{\mathrm{E}} = \kappa P^{\mathrm{dc}},$$

where κ is the unit cost per W of momentary power consumption.

The simplified electricity cost model is used in Papers III and V.

4.4 Optimization Scopes

Solutions to the problem of power budgeting can have various scopes. Some may be very selective and try to optimize only applications co-located on a single server during high workload periods, while others may tackle the problem by proposing a general power-budget-aware scheduler for entire data centers. Here we analyze the potential for optimization within the time and space scopes, and try to position the power budgeting approaches presented here in this two-dimensional space.

Time Scope

The most severe power budgeting issues occur during high-workload periods when the available power budget is too small to handle all the application requests. Although many application workloads are periodic and predictable, unpredictable workload spikes occur from time to time in practice. Therefore, it is natural to manage power budgeting issues reactively, using some kind of *emergency system* that kicks in when the power consumption becomes dangerously close to the power limit.

On the other hand, the probability of power budgeting incidents can be reduced through proactive optimization of data center infrastructures during normal operation. Actions such as application placement can be used by a *general scheduler* at the cluster level to avoid co-locating applications that are likely to consume a lot of power at the same time.

Finally, since power budgeting is something introduced primarily to reduce the capital expenditures of data center owners, an appropriate time to plan and evaluate applicable strategies is when *planning the data center's capacity*. The frequency of power budgeting incidents will depend primarily on decisions made during the planning phase. If the power delivery infrastructure design is aggressive and the data center's power limit is close to its usual power consumption range, it will make more sense to include power budgeting awareness in the general scheduling logic. However, if the design is conservative, power budgeting incidents should be rare and can be handled reactively by an emergency system.

The Power Budgeting Controller as an Emergency System

Power budgeting incidents happen when the total power consumption reaches a predefined budget limit. If such incidents are rare, it makes sense to handle them using a specialized mechanism that first identifies that an incident is happening and then reacts accordingly. Such an approach is used in Facebook's Dynamo [Wu+16] system, where the power budget controller becomes active only when the power consumption reaches a so-called *capping threshold*, typically set to 99% of the power limit. The power capping techniques are kept active until the workload (and thus the power consumption) falls to a safe level (typically set to a value below the activation threshold to avoid oscillations).

These kinds of controllers are reactive in nature and operate only when power becomes a scarce resource in the data center as a whole or in a smaller component such as a rack or cluster. This approach does not require the ability to predict application workload levels, but might benefit from incorporating models describing applications' power-performance tradeoffs that could be used to evaluate the impact of potential configurations before they are implemented.

Power-Performance Aware General Schedulers

In the general scheduler approach, all data center infrastructure optimization decisions, especially those relating to placement and scheduling, should account

for power-performance tradeoffs. Therefore, the scheduler requires considerable knowledge about the hosted applications and their behavior, including temporal workload level forecasts.

This approach makes it possible to address problems proactively and could potentially enable the use of a wider range of actuators than a reactive approach. On the other hand, there is also an argument against including such mechanisms in a general scheduler. Existing data center schedulers are already very complex systems that must consider multiple constraint types and optimization goals. Therefore, many data center operators might be reluctant to add additional features to their general schedulers that will only be needed in extreme cases.

Data Center Capacity Planning

When designing a new data center, the investor has to decide what level of peak power consumption the data center's power delivery infrastructure should be able to sustain. Because of the dynamic nature of cloud computing workloads and the overall costs of power delivery infrastructure, this decision is non-trivial. A conservative approach of delivering the theoretical maximum power consumption will increase the cost of the infrastructure, while an aggressive approach of underprovisioning power delivery infrastructure may make it difficult to ensure a high QoS during high workload periods.

Power budgeting strategies must also be considered when upgrading hardware to a newer generation, which usually increases infrastructure density and peak power consumption.

Space Scope

Cloud data center infrastructures are very complex and are therefore organized in a hierarchical way to facilitate their management. Additionally, power-performance tradeoffs in data center infrastructures can be managed using various techniques that are implemented at multiple infrastructure levels. Figure 4.6 shows various power budgeting approaches that can be applied at different levels of the data center infrastructure and their interactions. These approaches range from total power budget selection at the top level to cluster power shifting, server power capping, and graceful application degradation. Below, we describe each approach explaining its inputs and outputs, as well as some possible optimization methods.

Power Budget Selection

Power budget selection is the process of choosing an optimal limit on the total power consumption of a given part of the infrastructure. This approach is best implemented at the level of the whole data center, at which the infrastructure owner is billed for electricity.

Figure 4.7 visualizes the problem of power budget selection. The physical limit on power consumption that the infrastructure can sustain is P^{limit}.

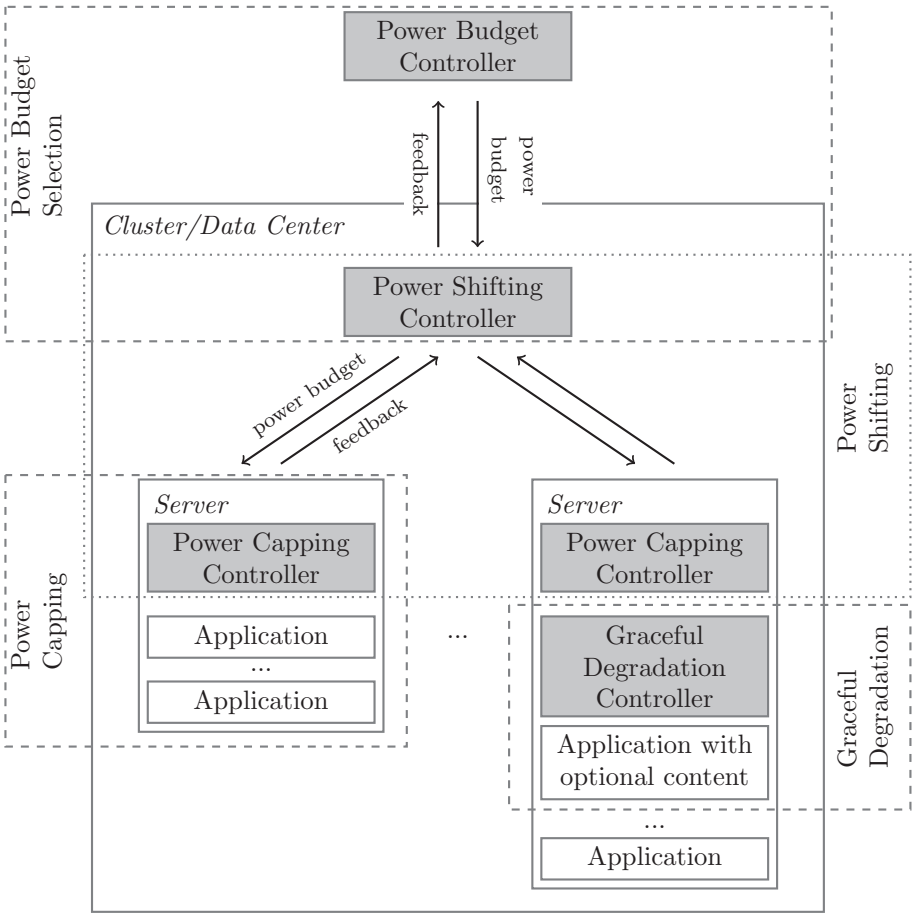

Figure 4.6: Power budgeting approaches, showing the major components of a cloud data center infrastructure and the space scope of each approach.

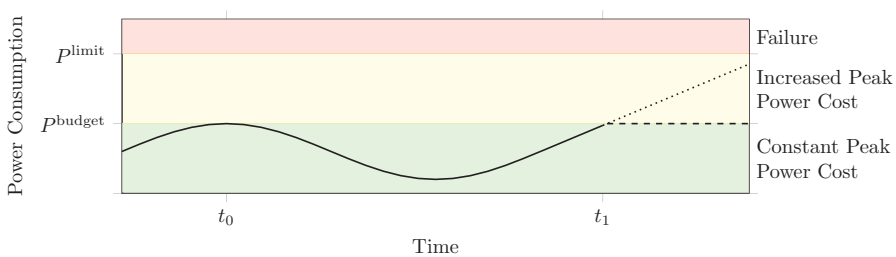

Figure 4.7: The data center power budget selection problem, showing key parameters and the consequences of possible decisions. At time t_1, the operator must decide whether the power consumption should be throttled to the level specified by the power budget (- -) or left unconstrained (····).

However, to reduce electricity costs, the data center operator may decide to operate with a lower power budget P^{budget}. At some point in time, t_1 in this example, the increasing power consumption approaches the previously defined power budget, P^{budget}. At this moment, the data center operator must decide whether to increase the power budget and risk increasing their electricity costs or enforce it and risk application performance degradation.

When selecting the operating power budget, the following factors should be considered:

- the maximum power consumption achieved in the current billing period (time t_0 in Figure 4.7), because exceeding this threshold will increase the peak power component of the electricity bill,
- the electricity pricing scheme, to quantify the costs of increased energy consumption and peak power cost,
- the total cost of penalties for application QoS violations, to quantify the costs of performance degradation due to limiting the power budget,
- the time left until the end of the billing period, to evaluate the possibility of another high workload occurring within the same period.

An output of the power budget controller, the selected power budget, is forwarded to the power shifting controller, which will decide how to divide it over the whole infrastructure.

Power Shifting

The power shifting approach allows power limits to be adjusted during the execution time. To make sense, this approach requires multiple servers; it can therefore be implemented at the rack, PDU, cluster, or even whole data center levels. It is also possible to incorporate multiple power shifting controllers into a hierarchical setup at several different levels.

The inputs of a power shifting controller should be:

- the power budget set by the power budget controller (or a higher level controller in a hierarchical setup),
- the utilization of server power budgets (or lower level components in a hierarchical setup),
- the performance of hosted applications (or an aggregated performance metric in a hierarchical setup).

The principal output of a power shifting controller is a new server power budget distribution (or a new power distribution for lower level components in a hierarchical setup). It should also provide aggregated feedback on QoS violations to the power budget controller.

Power Capping

Power capping involves imposing a limit on the power consumption of hardware components and is usually implemented at the level of a single server. Technologies commonly used for power capping include DVFS, CPU pinning [Coc+11], forced idleness [Gan+09a], and RAPL.

To minimize operational costs, a power capping controller should take the following factors into consideration:

- the server power budget decided by the power shifting controller,
- the performance of hosted applications,
- application QoS violation costs,
- electricity costs.

The power capping controller first decides whether it should utilize all of the available power budget or allow the power shifting controller to redistribute some of it to other servers. It then prioritizes the hosted applications to minimize total operational costs. It also provides feedback to the power shifting controller to facilitate the performance-awareness of higher level components.

Graceful Degradation

The graceful degradation approach facilitates the adaptation of application execution to the availability of resources and the power budget level. As an input, the graceful degradation controller receives power budgets for applications from the server power capping controller. It then internally tailors the way in which the application performs its tasks based on the available power, e.g., by reducing the percentage of requests served with optional content. Finally, the graceful degradation controller gives the server power capping controller feedback on the current degradation level and the potential for improvement in the event of an increase in the power budget.

Table 4.2: A summary of application performance aware optimization solutions proposed in this thesis

Approach	**Solution**	**Method**
Graceful degradation	Power-Aware Brownout	Feedback control
Power capping	ALPACA	Convex optimization
Power shifting	Power Shepherd	Convex optimization

Table 4.2 summarizes the approaches for managing power-performance tradeoffs presented in this thesis. In terms of the space-scope, we have examined

solutions acting at three different infrastructure levels. For the application level, we have proposed the Power-Aware Brownout controller (Paper IV). Paper III introduces ALPACA, a power capping framework operating at the server level. Finally, Paper V presents an investigation into power shifting at the cluster level. We also used two different methods to manage power-performance tradeoffs. Power-Aware Brownout is a cascade controller designed using control theoretical principles. Both the server- and cluster-level approaches use convex optimizers to minimize operational costs.

Chapter 5

Summary of Contributions

This thesis consists of six papers. The relationships between these papers are visualized in Figure 5.1. The initial two papers are more exploratory in terms of both the types of actuators that are examined and the range of server utilization levels that are analyzed, while Papers III–VI focus on narrower ranges of actuators and utilization levels. The ordering of the first three papers reflects a top-down approach in terms of the range of actuators that are utilized. Paper I has the broadest scope, covering the entire hierarchy of actuators (and the associated sensors) that are available in data centers. Paper II focuses on the subset of actuators used for server throttling and analyzes their power-performance tradeoffs over the full spectrum of server utilization levels. The remaining papers concentrate on situations in which power becomes a bottleneck in a data center, necessitating server throttling. However, they differ in their optimization scopes. Paper III focuses on the power capping of a single server with a given power budget and prioritization of co-located applications. Paper IV tackles the very specific case of applications with optional content that can adapt their computational needs to the availability of resources. Paper V extends the scope of optimization to a multi-server scenario, adding cluster-level power shifting but retaining an externally determined cluster power budget. Finally, Paper VI examines the problem of selecting an appropriate data-center-level power budget and presents a simulation framework for evaluating and comparing power budgeting strategies at multiple infrastructure levels.

The following sections describe the contributions presented in each paper and those made specifically by the author of this thesis. Erik Elmroth, Per-Olov Östberg (except for Paper IV), and Ahmed Ali-Eldin (except for Papers I and IV) played advisory roles in all of the studies described in these papers. Their contributions include discussions on problem formulation, methods, experiments, and presentation of results. They also provided feedback on the papers' content during the writing process.

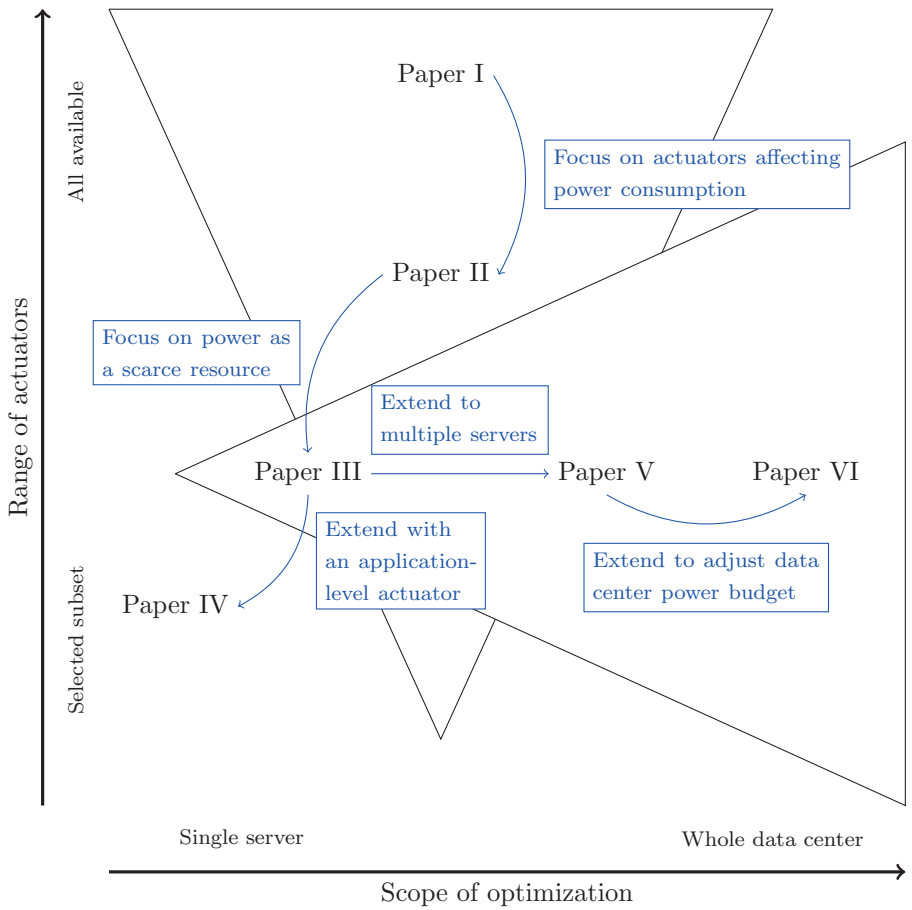

Figure 5.1: Scope of the included papers, showing the range of actuators considered in each case (from all available to selected subsets), the scope of the optimization in the space dimension (which ranges from single servers to the whole data center), and the relationships between the papers.

5.1 Paper I

J. Krzywda, P-O. Östberg, and E. Elmroth. A Sensor-Actuator Model for Data Center Optimization. *Technical Report UMINF 15.20*, Umeå University, Sweden, 2015.

This report is an extended version of a paper that appeared under the same title in *Proceedings of the 2015 IEEE International Conference on Cloud and Autonomic Computing (ICCAC)*, pp. 192–195, 2015.

Paper Contributions

In Paper I [KÖE15], we focus on **RO1** by addressing the topic of data center automation and proposing a sensor-actuator model for autonomic data center configuration optimization that is based on the MAPE-K loop for autonomic systems. The model characterizes various data center events (e.g., the arrival of a new virtual machine or a hardware failure), relates these events to a known set of actions that can be used to optimize data center operations (actuators), and identifies the data needed for this type of optimization and the timescales over which these data should be recorded. The paper then proposes taxonomies for selected aspects of data center configuration, elements of data center actuators, and classes of monitoring sensors. To support the analysis, the paper also characterizes and discusses the relationships between data center events, sensors, and actuators, and presents results from testbed experiments designed to illustrate selected tradeoffs.

On the basis of the analysis and experiments described above, data center events are classified into three distinct groups: planned, predicted, and unpredicted events. Each group is characterized. The paper also analyzes the spectrum of actuators available in data centers and the applicability and suitability of different actions for handling specific events. Finally, we discuss the types of data that should be monitored by sensors, and how frequently monitoring should be performed to facilitate detection and prediction of data center events.

Author's Contributions

The author of this thesis helped delineate the paper's scope and proposed taxonomies of events, sensors, and actuators. He also conducted testbed experiments analyzing selected actuators, and wrote the paper with feedback from the other co-authors.

5.2 Paper II

J. Krzywda, A. Ali-Eldin, T. E. Carlson, P-O. Östberg, and E. Elmroth. Power-Performance Tradeoffs in Data Center Servers: DVFS, CPU Pinning, Horizontal, and Vertical Scaling. *Future Generation Computer Systems*, Elsevier, Vol. 81, pp. 114–128, 2018.

Paper Contributions

In Paper II [Krz+18a], we narrow our focus to a subset of data center actuators that affect power consumption. To address **RO1**, we perform an in-depth analysis of four actuators that can be used to throttle servers: DVFS, CPU pinning, horizontal and vertical scaling. We also address **RO2** by modeling the effect of CPU pinning on server power consumption.

This paper shows that the impact of DVFS on the power consumption of underloaded servers is limited by the CPU idle states, and that for some request arrival patterns (e.g., bursty arrival patterns), reducing the CPU frequency of underloaded servers actually *increases* power consumption. Another notable finding is that consolidation of virtual CPUs using CPU pinning reduces power consumption at the cost of performance degradation. However, the application performance degradation due to consolidation of virtual CPUs can be limited by choosing an appropriate CPU pinning scheme. Moreover, we show that combining horizontal and vertical scaling with consolidation of virtual CPUs can reduce power consumption at high resource utilization levels. While the ability of horizontal and vertical scaling to improve response times and throughput is limited when physical servers are highly loaded, we show that it can be increased by CPU pinning. Finally, we observe that the load balancing strategy has a big impact on the tail response time of horizontally scaled applications.

These findings show that the power budget of data centers and the performance of hosted applications can be manipulated by adjusting the configuration of physical servers (DVFS), virtual machines (by implementing horizontal and vertical scaling), and the mapping between the two (CPU pinning). Based on these findings, we present a set of recommendations for using these actuators in ways that account for power-performance tradeoffs.

Author's Contributions

Based on his own observations from Paper I and discussions with others, the author identified the set of actuators to be evaluated (thereby defining the paper's scope) and developed the methodology for comparing their impacts on application performance and server power consumption. The author also designed and conducted the testbed experiments with feedback from others, and heavily extended scripts developed by others to suit the study's purposes. In addition, the author performed a statistical analysis of the results and wrote the entire paper, incorporating feedback provided by the other authors.

5.3 Paper III

J. Krzywda, A. Ali-Eldin, E. Wadbro, P-O. Östberg, and E. Elmroth. ALPACA: Application Performance Aware Server Power Capping. *Proceedings of the 15th IEEE International Conference on Autonomic Computing (ICAC 2018)*, pp. 41–50, 2018.

Paper Contributions

Paper III [Krz+18b] builds on the findings of Paper II and proposes a solution for managing applications hosted on a server with a limited power budget. We extend our work on **RO1** by analyzing RAPL technology and cgroups, address **RO2** by proposing application power-performance models, and tackle **RO3** by proposing a framework that minimizes application performance degradation under power constraints.

In this paper, we investigate ways of dividing the server power budget between the hosted applications and of co-locating applications on servers with limited power budgets so as to minimize application performance degradation. We propose the ALPACA framework, which has three main components: a model, an optimizer, and a controller. The application power-performance model captures the relationship between the power budget and application performance. The optimizer chooses the best combination of power budget levels for each application. The controller enforces the optimal application power budgets using RAPL, CPU quotas, and shares, and by suspending applications where appropriate.

We evaluate the proposed solution on a real testbed using four well-established cloud computing applications: MediaWiki, SysBench, Web Search from CloudSuite, and the Sock Shop microservice benchmark. ALPACA was found to reduce data centers' operational costs over the whole range of power budgets by reducing unnecessarily high power consumption when it provides no performance gain, and by switching off the most costly applications when power is scarce. Overall, ALPACA reduced operational costs by up to 20%.

Author's Contributions

The author was primarily responsible for formulating the problem addressed in this paper, and developed the methodology used to solve the problem with the assistance of the co-authors' feedback. He implemented all the components of the ALPACA framework, namely the automated capture of application models, the testbed controller, and the simulator. The author also contributed to the optimizer's development by helping with the formal formulation of the optimization problem and extending the MATLAB implementation of the optimization function. In addition, he designed and conducted all the testbed and simulation experiments, and analyzed their results. Finally, he wrote the entire paper with feedback from the other authors.

5.4 Paper IV

A. V. Papadopoulos, J. Krzywda, E. Elmroth, and M. Maggio. Power-Aware Cloud Brownout: Response Time and Power Consumption Control. *Proceedings of the 2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 2686–2691, 2017.

Paper Contributions

Paper IV [Pap+17] explores the possibility of combining power capping with brownout—an established application-level technique for response time control. We propose a novel application of the brownout concept, which reduces the resource consumption of an application by deactivating its optional functions in response to power budget shortages. The paper presents a design for a cascaded controller that ensures running applications adhere to a power budget by varying their response time set points to adjust the load they receive. The proposed solution is evaluated using a brownout simulator and a model of the RUBiS application captured on a testbed.

This work addresses **RO1** by investigating the scope for managing power-performance tradeoffs by adjusting the percentage of requests served with optional content. It also addresses **RO2** by extending the ALPACA application power-performance models to the brownout use case. Finally, it addresses **RO3** via the implementation of a cascade controller for managing application response times and server power consumption.

Author's Contributions

The author of this thesis and Alessandro V. Papadopoulos were considered equal contributors and hence joint first authors.

Based on his own experiences of optimizing various application types during the work on Paper III, the author developed the idea of using applications with optional computation tasks to extend the operational interval of power capping, and of using a brownout controller to minimize the cost of performance degradation (the main contributor to the problem formulation and scope of paper). He extended ALPACA's modeling component to capture the power-performance tradeoffs for applications with optional content, using RUBiS as a showcase application. He also helped write the paper by drafting the sections dealing with power capping (Section 2.2) and application benchmarking (Section 4.1).

5.5 Paper V

J. Krzywda, A. Ali-Eldin, E. Wadbro, P-O. Östberg, and E. Elmroth. Power Shepherd: Application Performance Aware Power Shifting. *Submitted*, 2019.

Paper Contributions

In Paper V, we tackle the problem of application performance aware power shifting between multiple servers. First, we investigate the potential for extending the ALPACA optimizer, proposed in Paper III for managing applications co-located on a single server, to handle multiple servers. The necessary extensions were implemented, but the scalability of the multi-server ALPACA optimizer proved to be unsatisfactory. We therefore changed our approach and developed an alternative hierarchical solution: Power Shepherd.

Power Shepherd consists of a cluster level controller for power shifting and a server level controller for power capping. This solution reduces the operational costs of data center clusters by directing power to servers where it is most needed. We implemented a prototype solution, evaluated it in a real testbed, assessed its scalability, and discussed its applicability and limitations.

This work addresses **RO1** by focusing on the power shifting approach and evaluating its usefulness. Moreover, we implemented a monitoring framework that allows us to collect metrics from multiple servers in real time. **RO2** is addressed by quantifying the influence of power shifting on data centers' operating costs when facing workload levels that are unevenly distributed between servers. Finally, **RO3** is addressed by the design and implementation of a cluster-level controller for power shifting and by extending the server controller from Paper III to cooperate in a multi-server environment.

Author's Contributions

The author of this thesis developed the overall idea of applying application performance awareness to power shifting. He was the main contributor to the problem's formulation, delineated the paper's scope, and proposed methodology with feedback from the co-authors. While working on the project, he implemented the cluster controller and extended the ALPACA controller. To facilitate experimentation, he implemented a testbed monitoring system (for details, see Section 5.8). He conducted experiments and analyzed their results. Finally, he wrote the entire paper, incorporating feedback from the other co-authors.

5.6 Paper VI

J. Krzywda, V. Meyer, M. G. Xavier, A. Ali-Eldin, P-O. Östberg, C. A. F. De Rose, and E. Elmroth. Modeling and Simulation of QoS-Aware Power Budgeting in Cloud Data Centers. *Submitted*, 2019.

Paper Contributions

Paper VI covers the whole spectrum of optimization scopes: from single server power capping to adjusting the power budget of a whole data center. This paper describes the extension of CloudSim, giving it the ability to simulate the power capping of servers equipped with Running Average Power Limit (RAPL) systems, power shifting between servers, and application performance degradation due to limited power budgets. We propose an optimization model to minimize data centers' operational costs by accounting for both QoS violations and monthly electricity costs, the latter of which depend on both the total energy consumed over the period and the peak power consumption.

This work addresses **RO1** by analyzing the actuator for adjusting the power budget of the whole data center. **RO2** was also addressed in this work by porting our application power-performance models into CloudSim and validating them. Finally, in relation to **RO3**, we combined three controllers into a global system for power budgeting over a whole data center.

Author's Contributions

The author was one of the originators of the overall idea of extending CloudSim to include RAPL capabilities. He delineated the paper's scope during discussions with other co-authors. He extended CloudSim with RAPL capabilities and application performance models with the help of other co-authors. He also participated in the design, implementation, and analysis of the experiments. Finally, he proposed the initial structure of the paper, which was subsequently refined in collaboration with other authors, and contributed significantly to the writing process.

5.7 Limitations

As noted in the introduction section, data center infrastructures are very complex, and this complexity is compounded when considering research problems relating to the optimization of infrastructure configurations. Therefore, to study specific phenomena, one must limit the scope of one's research and make certain assumptions.

In the work presented here, we chose to exclude some dimensions that were expected to complicate the problem so severely as to make progress unlikely. These choices may limit the direct applicability of some results presented here in certain real-world scenarios. Overcoming these limitations may require further investigations and extensions of the presented work. Here we summarize the most important limitations of our work (at least in our view), many of which reveal potentially interesting directions for future research.

Time Awareness

All the controllers and optimizers utilized in this thesis make decisions based only on the system's current state. They do not consider the history of previous states or any kinds of predictions.

However, accounting for the time dimension could be very beneficial for power budget management. First, including the history of states could help to avoid oscillations in configuration decisions. For example, the benefit of turning an application on and off multiple times over a short period because of changes in other application may be insignificant. Moreover, in the case of a long-running batch job, it may be sensible to let the application finish processing if its time-to-completion is small, even if that makes the system's configuration briefly non-optimal.

However, embedding the notion of history and changes over time into optimization problems significantly increases their complexity. Also, the prediction of application workload levels and execution times is a research topic in its own right.

Multi-Component Applications

In our work on power budgeting, we only considered assigning power budgets to whole applications. Most of the applications we examined in our experiments are monolithic: SysBench is a single binary, and MediaWiki was deployed on a single virtual machine. Even when we used the microservice architecture for Sock Shop, we treated all the containers as a single unit by assigning identical parameters to all of them.

In reality, applications can consist of several components, each of which could be optimized independently to achieve the best possible configuration. However, such an approach would require a detailed understanding of the inter-

dependencies between application components, including dynamic identification of components that are performance bottlenecks in specific situations.

5.8 Software Artifacts

As noted in the research methodology section (Section 1.2), this work used a constructive research method, which typically results in the production of artifacts. Below, we describe the software artifacts that we created to evaluate the ideas presented in this thesis. To increase the reproducibility of our research results and facilitate their extension or further evaluation, all of these artifacts have been open sourced.[1]

The ALPACA Modeling and Optimization Framework[2]

The first artifact is a framework we implemented while working on Paper III. The ALPACA framework has three components:

Modeller, which facilitates the capture of application power-performance models. These models describe the relationship between the power budget allocated to an application and its performance. We have automated the process of benchmarking applications and finding a function that describes the relationship as accurately as reasonably possible.

Optimizer, which minimizes a data center's operating costs for a given server power budget based on the application power-performance models generated with Modeller. The costs include costs due to Quality of Service violations and electricity costs.

Controller, which enforces the application power budgets determined by the optimizer. The controller uses various technologies available in modern servers, including RAPL for power capping and Linux CFS parameters.

Power-Performance Tradeoff Monitoring Systems

To facilitate the development and conduct of multi-server testbed experiments for Paper V, we designed and set up a monitoring framework using *Prometheus*[3] to collect metrics and *Grafana*[4] to display the metrics in real time while the experiments were in progress.

This framework was based on *Swarmprom*, a starter kit for Docker Swarm monitoring developed by Stefan Pordan.[5]

Our monitoring system uses the following exporters:

[1] `http://www8.cs.umu.se/wp/jakub/reproduce/`
[2] `https://git.cs.umu.se/jakub/alpaca-power-capping/`
[3] `https://prometheus.io/`
[4] `https://grafana.com/`
[5] `https://github.com/stefanprodan/swarmprom`

snmp-exporter, which collects power metrics from the intelligent PDU and exposes them to Prometheus.

node-exporter, which is used together with cron (a time-based job scheduler) to expose application workload and performance metrics.

During our experiments, we monitored the following metrics:

- at the server level:
  - RAPL power budget (per socket),
  - RAPL power consumption (per socket),
  - total power consumption (from the intelligent PDU),
- at the application level:
  - application state (running or not),
  - workload level,
  - CPU pinning,
  - CPU quota, CPU period, and CPU shares,
  - application-specific performance metrics.

By default, Prometheus does not provide persistent storage of collected metrics. Therefore, data stored by a standard Prometheus installation should be regarded as an ephemeral sliding window of recent results. To store experimental results for further analysis, we exported monitoring data in JSON format using the Prometheus HTTP API and transformed them into CSV format using *jq*.[6]

Power Shepherd Optimizer and Controllers[7]

Power Shepherd is a power shifting system that is described and evaluated in Paper V. It consists of two controllers:

Cluster controller, which allocates power budgets to servers according to the offers generated by server controllers. Each offer predicts the Quality of Service violation costs of applications hosted on a particular server for a given power budget. Based on the offers, the cluster controller, tries to reduce the power budget of servers that will suffer less and to shift power towards servers that will benefit the most.

Server controller, which prioritizes applications co-located on a server to minimize their performance degradation. Based on the power-performance models of hosted applications it predicts server operating costs for increased and reduced server power budgets, and exposes them to the cluster controller.

[6] `https://stedolan.github.io/jq/`

[7] `https://git.cs.umu.se/jakub/power-shepherd`

Controllers of both types continuously communicate with each other, iteratively adjust each server's power budget, and thereby minimize overall data center operating costs.

We used the Spring Framework[8] as a skeleton for the controllers. To facilitate sharing data among controllers, we used the Apache Cassandra database.[9]

CloudSim Power Budgeting Extension[10]

Paper VI proposes a framework for simulating power budgeting in cloud data centers. We implemented the proposed framework as an extension to an existing simulator—CloudSim [Cal+11].

CloudSim is a well established and popular cloud infrastructure simulator and has been upgraded with multiple features such as power models [BB12], DVFS modeling [Gué+13], and ACPI Global/Sleep states [Xav+17]. As a basis for our extension, we used the DVFS modeling version. Details of our extension are presented in Section 4 of Paper VI.

5.9 Connection to Research Objectives

This section recalls the main research objectives of this thesis, which were introduced in Section 1.1, and explains how each paper contributed towards achieving them.

The research objectives were:

RO1 To assess the usefulness of various metrics (sensors) and software configuration techniques (actuators) for controlling data center infrastructures.

RO2 To quantify and model the influence of actuators on server power consumption and application performance.

RO3 To develop controllers that optimize the configuration of data center components to achieve a predefined power-performance goal.

Table 5.1 summarizes the contributions of each paper towards achieving each research objective.

[8] `https://spring.io/`
[9] `http://cassandra.apache.org/`
[10] `https://git.cs.umu.se/jakub/cloudsim-powerbudgeting`

Table 5.1: Summary of individual papers' contributions towards the research objectives

	RO1: Assess sensors and actuators	**RO2**: Model power-performance tradeoffs	**RO3**: Develop controllers
Paper I	Created taxonomies for selected aspects of data center configuration, elements of data center actuators, and classes of monitoring sensors.	—	—
Paper II	Analyzed applicability of DVFS, CPU pinning, and horizontal and vertical scaling at various levels of server utilization.	Modeled the relationship between the number of utilized CPU cores and server power consumption with and without CPU pinning.	—
Paper III	RAPL for power capping and cgroups (CPU quota/period and CPU shares) for application performance isolation.	Proposed the ALPACA application power-performance model and captured instances for multiple cloud applications.	A single server controller to minimize momentary data center operating costs.
Paper IV	Application-level actuator (percentage of requests served with optional content), and the use of response time and power consumption as metrics.	Captured an ALPACA model for an application with optional content (the RUBiS benchmark).	A cascade controller to simultaneously manage application response time and power consumption.
Paper V	Power shifting to distribute the available power budget among servers.	Quantified tradeoffs between operational costs and power budget violations in clusters with unbalanced workloads.	A distributed system with a single power shifting controller and multiple server power capping controllers to minimize momentary data center operational costs.
Paper VI	Adjusting power budget of the whole data center.	Modeled application power-performance tradeoffs in CloudSim, simulated and validated against testbed runs.	A system comprising a power budget selection controller, a power shifting controller, and multiple server power capping controllers to minimize data center operational costs over a billing period.

Bibliography

[Asc18] Rhonda Ascierto. *Uptime Institute Global Data Center Survey*. 2018. URL: https://uptimeinstitute.com/2018-data-center-industry-survey-results (visited on May 24, 2019).

[BAB12] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing". In: *Future Generation Computer Systems* 28.5 (2012). Special Section: Energy efficiency in large-scale distributed systems, pp. 755–768. ISSN: 0167-739X. DOI: 10.1016/j.future.2011.04.017.

[Baw16] Tom Bawden. *Global warming: Data centres to consume three times as much energy in next decade, experts warn*. 2016. URL: http://www.independent.co.uk/environment/global-warming-data-centres-to-consume-three-times-as-much-energy-in-next-decade-experts-warn-a6830086.html (visited on June 2, 2019).

[BB10] Anton Beloglazov and Rajkumar Buyya. "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers". In: *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*. MGC '10. Bangalore, India: ACM, 2010, 4:1–4:6. ISBN: 978-1-4503-0453-5. DOI: 10.1145/1890799.1890803.

[BB12] Anton Beloglazov and Rajkumar Buyya. "Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers". In: *Concurrency and Computation: Practise and Experience* 24.13 (Sept. 2012), pp. 1397–1420. ISSN: 1532-0626. DOI: 10.1002/cpe.1867.

[BDH03] Luiz A. Barroso, Jeffrey Dean, and Urs Hölzle. "Web search for a planet: The Google cluster architecture". In: *IEEE Micro* 23.2 (Mar. 2003), pp. 22–28. ISSN: 0272-1732. DOI: 10.1109/MM.2003.1196112.

[Bea+09] John Bean, Ron Bednar, Richard Jones, Robb Jones, Phil Morris, David Moss, Michael Patterson, Joe Prisco, Wade Vinson, and John Wallerich. *Proper Sizing of IT Power and Cooling Loads*. Tech. rep. White Paper #23. The Green Grid, July 2009. URL: https://www.thegreengrid.org/en/resources/library-and-tools/392-WP#23---Proper-Sizing-of-IT-Power-and-Cooling-Loads-.

[BH07] Luiz A. Barroso and Urs Hölzle. "The Case for Energy-Proportional Computing". In: *Computer* 40.12 (2007), pp. 33–37. ISSN: 0018-9162. DOI: 10.1109/MC.2007.443.

[BHR18] Luiz A. Barroso, Urs Hölzle, and Parthasarathy Ranganathan. "The Datacenter as a Computer: Designing Warehouse-Scale Machines". In: *Synthesis Lectures on Computer Architecture* 13.3 (2018), pp. 1–189. DOI: 10.2200/S00874ED3V01Y201809CAC046.

[Cal+11] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms". In: *Software: Practice and Experience* 41.1 (2011), pp. 23–50. DOI: 10.1002/spe.995.

[CGJ97] Edward G. Coffman Jr, Michael R. Garey, and David S. Johnson. "Approximation Algorithms for NP-hard Problems". In: ed. by Dorit S. Hochbaum. Boston, MA, USA: PWS Publishing Co., 1997. Chap. Approximation Algorithms for Bin Packing: A Survey, pp. 46–93. ISBN: 0-534-94968-1. URL: http://dl.acm.org/citation.cfm?id=241938.241940.

[Coc+11] Ryan Cochran, Can Hankendi, Ayse K. Coskun, and Sherief Reda. "Pack & Cap: Adaptive DVFS and Thread Packing Under Power Caps". In: *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*. MICRO-44. Porto Alegre, Brazil: ACM, 2011, pp. 175–185. ISBN: 978-1-4503-1053-6. DOI: 10.1145/2155620.2155641.

[Crn10] Gordana D. Crnkovic. "Constructive Research and Info-computational Knowledge Generation". In: *Model-Based Reasoning in Science and Technology: Abduction, Logic, and Computational Discovery*. Ed. by Lorenzo Magnani, Walter Carnielli, and Claudio Pizzi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 359–380. ISBN: 978-3-642-15223-8. DOI: 10.1007/978-3-642-15223-8_20.

[Dav+10] Howard David, Eugene Gorbatov, Ulf R. Hanebutte, Rahul Khanna, and Christian Le. "RAPL: Memory power estimation and capping". In: *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*. Aug. 2010, pp. 189–194. DOI: 10.1145/1840845.1840883.

[Dós07] György Dósa. "The tight bound of first fit decreasing bin-packing algorithm is FFD (I) ≤ 11/9OPT (I)+ 6/9". In: *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–11. ISBN: 978-3-540-74450-4. DOI: 10.1007/978-3-540-74450-4_1.

[EG16] Rich Evans and Jim Gao. *DeepMind AI reduces energy used for cooling Google data centers by 40%*. 2016. URL: https://blog.google/outreach-initiatives/environment/deepmind-ai-reduces-energy-used-for/ (visited on May 24, 2019).

[Fac19a] Facebook. *Prineville Data Center*. 2019. URL: https://www.facebook.com/PrinevilleDataCenter/app/399244020173259/ (visited on May 24, 2019).

[Fac19b] Facebook. *Sustainability in numbers*. 2019. URL: https://www.sustainability.fb.com/sustainability-in-numbers/ (visited on May 24, 2019).

[FF05] Mark E. Femal and Vincent W. Freeh. "Boosting Data Center Performance Through Non-Uniform Power Allocation". In: *Second International Conference on Autonomic Computing (ICAC'05)*. June 2005, pp. 250–261. DOI: 10.1109/ICAC.2005.17.

[FWB07] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz A. Barroso. "Power Provisioning for a Warehouse-sized Computer". In: *Proceedings of the 34th Annual International Symposium on Computer Architecture*. ISCA '07. San Diego, California, USA: ACM, 2007, pp. 13–23. ISBN: 978-1-59593-706-3. DOI: 10.1145/1250662.1250665.

[Gan+09a] Anshul Gandhi, Mor Harchol-Balter, Rajarshi Das, Jeffrey O. Kephart, and Charles Lefurgy. "Power capping via forced idleness". In: *Workshop on Energy Efficient Design*. 2009, pp. 1–6. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.151.8070.

[Gan+09b] Anshul Gandhi, Mor Harchol-Balter, Rajarshi Das, and Charles Lefurgy. "Optimal Power Allocation in Server Farms". In: *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems*. SIGMETRICS '09. Seattle, WA, USA: ACM, 2009, pp. 157–168. ISBN: 978-1-60558-511-6. DOI: 10.1145/1555349.1555368.

[Gen19] Generac Power Systems. *Backup Power for Data Centers*. 2019. URL: http://www.generac.com/Industrial/industrial-solutions/data-centers (visited on June 2, 2019).

[Goo19] Google. *Data Centers, Efficiency: How we do it*. 2019. URL: https://www.google.com/about/datacenters/efficiency/internal/ (visited on May 24, 2019).

[GSU11] Sriram Govindan, Anand Sivasubramaniam, and Bhuvan Urgaonkar. "Benefits and Limitations of Tapping into Stored Energy for Datacenters". In: *Proceedings of the 38th Annual International Symposium on Computer Architecture*. ISCA '11. San Jose, California, USA: ACM, 2011, pp. 341–352. ISBN: 978-1-4503-0472-6. DOI: 10.1145/2000064.2000105.

[GSW13] Chang Ge, Zhili Sun, and Ning Wang. "A Survey of Power-Saving Techniques on Data Centers and Content Delivery Networks". In: *IEEE Communications Surveys Tutorials* 15.3 (Mar. 2013), pp. 1334–1354. ISSN: 1553-877X. DOI: 10.1109/SURV.2012.102512.00019.

[Gué+13] Tom Guérout, Thierry Monteil, Georges Da Costa, Rodrigo N. Calheiros, Rajkumar Buyya, and Mihai Alexandru. "Energy-aware simulation with DVFS". In: *Simulation Modelling Practice and Theory* 39 (2013). Special Issue on Energy efficiency in grids and clouds, pp. 76–91. ISSN: 1569-190X. DOI: 10.1016/j.simpat.2013.04.007.

[Har13] Mor Harchol-Balter. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, 2013. DOI: 10.1017/CBO9781139226424.

[HKH09] Jan Holmström, Mikko Ketokivi, and Ari-Pekka Hameri. "Bridging Practice and Theory: A Design Science Approach". In: *Decision Sciences* 40.1 (2009), pp. 65–87. DOI: 10.1111/j.1540-5915.2008.00221.x.

[HM14] Henry Hoffmann and Martina Maggio. "PCP: A Generalized Approach to Optimizing Performance Under Power Constraints through Resource Management". In: *11th International Conference on Autonomic Computing (ICAC 14)*. Philadelphia, PA: USENIX Association, June 2014, pp. 241–247. ISBN: 978-1-931971-11-9. URL: http://www.usenix.org/conference/icac14/technical-sessions/presentation/hoffman.

[Hua+11] Qiang Huang, Fengqian Gao, Rui Wang, and Zhengwei Qi. "Power Consumption of Virtual Machine Live Migration in Clouds". In: *2011 Third International Conference on Communications and Mobile Computing (CMC)*. Mar. 2011, pp. 122–125. DOI: 10.1109/CMC.2011.62.

[Jai91] Raj Jain. *The Art of Computer Systems Performance Analysis: Techniques For Experimental Design, Measurement, Simulation, and Modeling*. USA: Wiley, Apr. 1991. ISBN: 978-0-471-50336-1.

[Jeo+13] Jinkyu Jeong, Sung-Hun Kim, Hwanju Kim, Joonwon Lee, and Euiseong Seo. "Analysis of virtual machine live-migration as a method for power-capping". In: *The Journal of Supercomputing* 66.3 (2013), pp. 1629–1655. ISSN: 1573-0484. DOI: 10.1007/s11227-013-0956-1.

[JS15] Brendan Jennings and Rolf Stadler. "Resource Management in Clouds: Survey and Research Challenges". In: *Journal of Network and Systems Management* 23.3 (July 2015), pp. 567–619. ISSN: 1573-7705. DOI: 10.1007/s10922-014-9307-7.

[Kan+14] Svilen Kanev, Kim Hazelwood, Gu-Yeon Wei, and David Brooks. "Tradeoffs between Power Management and Tail Latency in Warehouse-Scale Applications". In: *2014 IEEE International Symposium on Workload Characterization (IISWC)*. Oct. 2014, pp. 31–40. DOI: 10.1109/IISWC.2014.6983037.

[KÖE15] Jakub Krzywda, Per-Olov Östberg, and Erik Elmroth. "A Sensor-Actuator Model for Data Center Optimization". In: *2015 International Conference on Cloud and Autonomic Computing*. Sept. 2015, pp. 192–195. DOI: 10.1109/ICCAC.2015.13.

[Kom+13] Toshiya Komoda, Shingo Hayashi, Takashi Nakada, Shinobu Miwa, and Hiroshi Nakamura. "Power Capping of CPU-GPU Heterogeneous Systems through Coordinating DVFS and Task Mapping". In: *2013 IEEE 31st International Conference on Computer Design (ICCD)*. Oct. 2013, pp. 349–356. DOI: 10.1109/ICCD.2013.6657064.

[Krz+15] Jakub Krzywda, Ahmed Ali-Eldin, Per-Olov Östberg, Ali Rezaie, Zafeirios Papazachos, Barry McCollum, Ryan Hamilton-Bryce, and Jörg Domaschka. *Context-Aware Cloud Topology Optimisation and Simulation: Extended Optimization Model*. 2015. URL: https://oparu.uni-ulm.de/xmlui/handle/123456789/4346 (visited on June 6, 2019).

[Krz+18a] Jakub Krzywda, Ahmed Ali-Eldin, Trevor E. Carlson, Per-Olov Östberg, and Erik Elmroth. "Power-performance tradeoffs in data center servers: DVFS, CPU pinning, horizontal, and vertical scaling". In: *Future Generation Computer Systems* 81 (2018), pp. 114–128. ISSN: 0167-739X. DOI: 10.1016/j.future.2017.10.044.

[Krz+18b] Jakub Krzywda, Ahmed Ali-Eldin, Eddie Wadbro, Per-Olov Östberg, and Erik Elmroth. "ALPACA: Application Performance Aware Server Power Capping". In: *the 15th IEEE International Conference on Autonomic Computing*. ICAC 2018. Sept. 2018, pp. 41–50. DOI: 10.1109/ICAC.2018.00014.

[Liu+11] Haikun Liu, Hai Jin, Cheng-Zhong Xu, and Xiaofei Liao. “Performance and energy modeling for live migration of virtual machines”. In: *Cluster Computing* 16.2 (2011), pp. 249–264. ISSN: 1573-7543. DOI: 10.1007/s10586-011-0194-3.

[Liu+16] Yanpei Liu, Guilherme Cox, Qingyuan Deng, Stark C. Draper, and Ricardo Bianchini. “FastCap: An efficient and fair algorithm for power capping in many-core systems”. In: *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. Apr. 2016, pp. 57–68. DOI: 10.1109/ISPASS.2016.7482074.

[Lo+14] David Lo, Liqun Cheng, Rama Govindaraju, Luiz A. Barroso, and Christos Kozyrakis. “Towards Energy Proportionality for Large-scale Latency-critical Workloads”. In: *Proceeding of the 41st Annual International Symposium on Computer Architecuture*. ISCA ’14. Minneapolis, Minnesota, USA: IEEE Press, 2014, pp. 301–312. ISBN: 978-1-4799-4394-4. URL: http://dl.acm.org/citation.cfm?id=2665671.2665718.

[LWW08] Charles Lefurgy, Xiaorui Wang, and Malcolm Ware. “Power capping: a prelude to power shifting”. In: *Cluster Computing* 11.2 (Jan. 2008), pp. 183–195. ISSN: 1573-7543. DOI: 10.1007/s10586-007-0045-4.

[Man15] Zoltán Ádám Mann. “Allocation of Virtual Machines in Cloud Data Centers—A Survey of Problem Models and Optimization Algorithms”. In: *ACM Computing Surveys (CSUR)* 48.1 (Aug. 2015), 11:1–11:34. ISSN: 0360-0300. DOI: 10.1145/2797211.

[Mei+11] David Meisner, Christopher M. Sadler, Luiz A. Barroso, Wolf-Dietrich Weber, and Thomas F. Wenisch. “Power management of online data-intensive services”. In: *Proceedings of the 38th Annual International Symposium on Computer Architecture*. ISCA ’11. June 2011, pp. 319–330. DOI: 10.1145/2000064.2000103.

[MGW09] David Meisner, Brian T. Gold, and Thomas F. Wenisch. “PowerNap: Eliminating Server Idle Power”. In: *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS XIV. Washington, DC, USA: ACM, 2009, pp. 205–216. ISBN: 978-1-60558-406-5. DOI: 10.1145/1508244.1508269.

[Mol82] Michael K. Molloy. “Performance Analysis Using Stochastic Petri Nets”. In: *IEEE Transactions on Computers* 31.09 (Sept. 1982), pp. 913–917. ISSN: 0018-9340. DOI: 10.1109/TC.1982.1676110.

[Mus+15] Saad Mustafa, Kashif Bilal, Sajjad A. Madani, Nikos Tziritas, Samee U. Khan, and Laurence T. Yang. "Performance Evaluation of Energy-Aware Best Fit Decreasing Algorithms for Cloud Environments". In: *2015 IEEE International Conference on Data Science and Data Intensive Systems*. Dec. 2015, pp. 464–469. DOI: 10.1109/DSDIS.2015.104.

[OAL14] Anne-Cecile Orgerie, Marcos D. Assunção, and Laurent Lefevre. "A Survey on Techniques for Improving the Energy Efficiency of Large-scale Distributed Systems". In: *ACM Computing Surveys* 46.4 (2014), 47:1–47:31. ISSN: 0360-0300. DOI: 10.1145/2532637.

[Öst+14] Per-Olov Östberg, Henning Groenda, Stefan Wesner, James Byrne, Dimitrios S. Nikolopoulos, Craig Sheridan, Jakub Krzywda, Ahmed Ali-Eldin, Johan Tordsson, Erik Elmroth, Christian Stier, Klaus Krogmann, Jörg Domaschka, Christopher B. Hauser, P. J. Byrne, Sergej Svorobej, Barry Mccollum, Zafeirios Papazachos, Darren Whigham, Stephan Rüth, and Dragana Paurevic. "The CACTOS Vision of Context-Aware Cloud Topology Optimization and Simulation". In: *Proceedings of the 6th IEEE International Conference on Cloud Computing Technology and Science*. CloudCom 2014. Dec. 2014, pp. 26–31. DOI: 10.1109/CloudCom.2014.62.

[OW2+99] OW2 Consortium et al. *RUBiS: Rice University Bidding System*. 1999. URL: http://rubis.ow2.org (visited on June 6, 2019).

[Pap+17] Alessandro V. Papadopoulos, Jakub Krzywda, Erik Elmroth, and Martina Maggio. "Power-aware cloud brownout: Response time and power consumption control". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. Dec. 2017, pp. 2686–2691. DOI: 10.1109/CDC.2017.8264049.

[Pod+15] Andrej Podzimek, Lubomír Bulej, Lydia Y. Chen, Walter Binder, and Petr Tuma. "Analyzing the Impact of CPU Pinning and Partial CPU Loads on Performance and Energy Efficiency". In: *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. May 2015, pp. 1–10. DOI: 10.1109/CCGrid.2015.164.

[QM11] Haiyang Qian and Deep Medhi. "Server Operational Cost Optimization for Cloud Computing Service Providers over a Time Horizon". In: *Proceedings of the 11th USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*. USENIX Association. 2011, pp. 1–6.

[Ran+06] Parthasarathy Ranganathan, Phil Leech, David Irwin, and Jeffrey Chase. "Ensemble-level Power Management for Dense Blade Servers". In: *Proceedings of the 33rd Annual International Symposium on Computer Architecture*. ISCA '06. Washington, DC, USA:

IEEE Computer Society, 2006, pp. 66–77. ISBN: 0-7695-2608-X. DOI: 10.1109/ISCA.2006.20.

[RPC08] Andy Rawson, John Pfleuger, and Tahir Cader. *Green Grid data center power efficiency metrics: PUE and DCiE*. Tech. rep. White Paper #6. The Green Grid, 2008.

[She+16] Arman Shehabi, Sarah J. Smith, Dale A. Sartor, Richard E. Brown, Magnus Herrlin, Jonathan G. Koomey, Eric R. Masanet, Nathaniel Horner, Inês L. Azevedo, and William Lintner. *United States Data Center Energy Usage Report*. 2016. URL: https://eta.lbl.gov/publications/united-states-data-center-energy (visited on June 2, 2019).

[SLT09] Mark Saunders, Philip Lewis, and Adrian Thornhill. "Understanding research philosophies and approaches". In: *Research methods for business students* 4 (2009), pp. 106–135.

[Svä+15] Petter Svärd, Benoit Hudzia, Steve Walsh, Johan Tordsson, and Erik Elmroth. "Principles and Performance Characteristics of Algorithms for Live VM Migration". In: *SIGOPS Operating Systems Review* 49.1 (Jan. 2015), pp. 142–155. ISSN: 0163-5980. DOI: 10.1145/2723872.2723894.

[Tay18] Yong Chiang Tay. *Analytical Performance Modeling for Computer Systems, Third Edition*. Morgan & Claypool, 2018. ISBN: 9781681733890. DOI: 10.2200/S00859ED3V01Y201806CSL010.

[TSB06] W. Pitt Turner IV, John H. Seader, and Kenneth G. Brill. "Tier classification define site infrastructure performance". In: *Uptime Institute* 17 (2006).

[TWT16] Selome Kostentinos Tesfatsion, Eddie Wadbro, and Johan Tordsson. "Autonomic Resource Management for Optimized Power and Performance in Multi-tenant Clouds". In: *2016 IEEE International Conference on Autonomic Computing (ICAC)*. July 2016, pp. 85–94. DOI: 10.1109/ICAC.2016.32.

[VKW14] Abhishek Verma, Madhukar Korupolu, and John Wilkes. "Evaluating job packing in warehouse-scale computing". In: *2014 IEEE International Conference on Cluster Computing (CLUSTER)*. Sept. 2014, pp. 48–56. DOI: 10.1109/CLUSTER.2014.6968735.

[Voo+09] William Voorsluys, James Broberg, Srikumar Venugopal, and Rajkumar Buyya. "Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings". In: ed. by Martin Gilje Jaatun, Gansen Zhao, and Chunming Rong. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. Chap. Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation, pp. 254–265. ISBN: 978-3-642-10665-1. DOI: 10.1007/978-3-642-10665-1_23.

[Wan+12] Xiaorui Wang, Ming Chen, Charles Lefurgy, and Tom W. Keller. "SHIP: A Scalable Hierarchical Power Control Architecture for Large-Scale Data Centers". In: *IEEE Transactions on Parallel and Distributed Systems* 23.1 (Jan. 2012), pp. 168–176. ISSN: 1045-9219. DOI: 10.1109/TPDS.2011.93.

[WCF10] Xiaorui Wang, Ming Chen, and Xing Fu. "MIMO Power Control for High-Density Servers in an Enclosure". In: *IEEE Transactions on Parallel and Distributed Systems* 21.10 (Oct. 2010), pp. 1412–1426. ISSN: 1045-9219. DOI: 10.1109/TPDS.2010.31.

[Wu+16] Qiang Wu, Qingyuan Deng, Lakshmi Ganesh, Chang-Hong Raymond Hsu, Yun Jin, Sanjeev Kumar, Bin Li, Justin Meza, and Yee Jiun Song. "Dynamo: Facebook's Data Center-Wide Power Management System". In: *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. June 2016, pp. 469–480. DOI: 10.1109/ISCA.2016.48.

[WW11] Xiaorui Wang and Yefu Wang. "Coordinating Power Control and Performance Management for Virtualized Server Clusters". In: *IEEE Transactions on Parallel and Distributed Systems* 22.2 (Feb. 2011), pp. 245–259. ISSN: 1045-9219. DOI: 10.1109/TPDS.2010.91.

[Xav+17] Miguel G. Xavier, Fábio D. Rossi, César A. F. De Rose, Rodrigo N. Calheiros., and Danielo G. Gomes. "Modeling and simulation of global and sleep states in ACPI-compliant energy-efficient cloud environments". In: *Concurrency and Computation: Practise and Experience* 29.4 (2017). DOI: 10.1002/cpe.3839.

[Yas14] Ahmad Yasin. "A Top-Down method for performance analysis and counters architecture". In: *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. Mar. 2014, pp. 35–44. DOI: 10.1109/ISPASS.2014.6844459.

[ZH15] Huazhe Zhang and Henry Hoffmann. "A Quantitative Evaluation of the RAPL Power Control System". In: *Feedback Computing* (2015).

[ZH16] Huazhe Zhang and Henry Hoffmann. "Maximizing Performance Under a Power Cap: A Comparison of Hardware, Software, and Hybrid Techniques". In: *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS '16. Atlanta, Georgia, USA: ACM, 2016, pp. 545–559. ISBN: 978-1-4503-4091-5. DOI: 10.1145/2872362.2872375.