FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Energy Efficient Cloud Control and Pricing in Geographically Distributed Data Centers

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Technischen Wissenschaften

eingereicht von

Dražen Lučanin
Matrikelnummer 0848810

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Univ. Prof. Dr. Ivona Brandić

Diese Dissertation haben begutachtet:

Univ. Prof. Dr. Ivona Brandić	Univ. Prof. Dr. Helmut Hlavacs

Wien, 21. April 2016

Dražen Lučanin

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Energy Efficient Cloud Control and Pricing in Geographically Distributed Data Centers

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Dražen Lučanin
Registration Number 0848810

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Univ. Prof. Dr. Ivona Brandić

The dissertation has been reviewed by:

Univ. Prof. Dr. Ivona Brandić	Univ. Prof. Dr. Helmut Hlavacs

Vienna, 21st April, 2016

Dražen Lučanin

Erklärung zur Verfassung der Arbeit

Dražen Lučanin
Bürgerspitalgasse 22/15, 1060 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 21. April 2016

Dražen Lučanin

Acknowledgements

The research leading to this thesis has received funding from the following projects: (1) the "Holistic Energy Efficient Approach for the Management of Hybrid Clouds" (HALEY) project, granted under the Vienna University of Technology research award; (2) several international meeting and training school grants by COST Action IC804 "Energy efficiency in large scale distributed systems"; (3) the short term scientific mission (STSM) grant by COST Action IC1305 "Network for Sustainable Ultrascale Computing" (NESUS).

I would like to thank my supervisor Univ. Prof. Dr. Ivona Brandić for her support and mentoring during my research at the Vienna University of Technology. I would also like to thank Univ. Prof. Dr. Helmut Hlavacs for providing feedback as an external reviewer of this thesis. Additionally, I would like to thank Senior Lecturer Dr. Rizos Sakellariou for our research collaboration and for hosting my research visit at the University of Manchester. I am also grateful to my research collaborators Ilia Pietri, Simon Holmbacka and Foued Jrad for their help in our joint work.

Big thanks go to my close colleagues and collaborators Soodeh Farkohi, Toni Mastelić, Ivan Brešković, Michael Maurer and Vincent Chimaobi Emeakaroha for the discussions, talks and fun we had during our period of working together. Additionally, I would like to thank the other members of the research groups I have had the privilege of collaborating with – the Distributed Systems Group and the Electronic Commerce Group at the Vienna University of Technology and the Division of Electronics at the Ruđer Bošković Institute in Zagreb, Croatia.

Finally, I would like to thank my partner Sara, my family and my friends who were always there for me when I needed advice or encouragement. I also have to thank my favorite musicians and bands who created the many great songs that were a musical inspiration during my work.

Dražen Lučanin
April 2016

Abstract

The fast pace of progress in the domain of applications provided over the Internet has created a need for computational resources delivered as an on-demand utility, without having to manually manage computer hardware. Cloud computing has emerged as a very popular paradigm where resources such as virtual machines are provided as a scalable, pay-as-you-go service, catering to applications in a multitude of fields.

On the other hand, the rapid cloud computing growth has turned the energy consumption of data centers hosting the cloud's hardware infrastructure into a global environmental problem and a major cost factor. It is estimated that data centers constitute 1.5% of global electricity usage. At the same time, to serve increasing user requirements, modern cloud providers are operating multiple geographically distributed data centers. Distributed data center infrastructure changes the rules of cloud control, as energy costs depend on current regional electricity prices and temperatures that we call geotemporal inputs. Furthermore, pricing policies at which cloud providers can offer computational resources depend on the quality of service (QoS). With such pricing schemes and the increasing energy costs in data centres, balancing energy savings with performance and revenue losses is a challenging problem for cloud providers. Existing cloud control methods are suitable only for a single data center or do not consider all the available cloud control actions that can reduce energy costs in geographically distributed data centers.

In this thesis, we propose a pervasive cloud control approach consisting of multiple methods for dynamic resource reallocation and hardware configuration adapted to volatile geotemporal inputs. The proposed methods consider the QoS impact of cloud control actions and the data quality limits of time series forecasting methods. We offer a cloud controller design that supports future extensions when new decision support components need to be added. We also propose novel pricing schemes which account for the computational resource availability and costs that arise from our cloud control approach to enable both flexible, energy-aware and high performance cloud computing.

We evaluate our cloud control methods empirically and in a number of simulations using historical traces of electricity prices, temperatures, workloads and other data. We estimate the potential energy cost savings by comparing our methods to state-of-the-art baseline methods. We explore a variety of input parameters to provide a range of guidelines for practical application of our methods in cloud systems. Our results show that significant energy cost savings are possible without harming the QoS or service revenue in geographically distributed cloud computing.

Kurzfassung

Der schnelle Fortschritt im Bereich von Internet-Anwendungen hat es erforderlich gemacht, Rechenressourcen on-demand zur Verfügung zu stellen ohne Computer-Hardware manuell managen zu müssen. Cloud Computing hat sich als sehr beliebtes Paradigma herauskristallisiert, um Ressourcen wie virtuelle Maschinen mannigfaltigen Anwendungen skalierbar in Form eines "Pay-as-you-goServices zugänglich zu machen.

Andererseits hat das rapide Wachstum von Cloud Computing den Energieverbrauch von Datenzentren, die die Cloud-Hardware betreiben, zu einem globalen Umweltproblem und einem gewichtigen Kostenfaktor gemacht. Man schätzt, dass Datenzentren 1,5% des globalen Elektrizitätsverbrauchs ausmachen. Gleichzeitig betreiben moderne Cloud-Provider mehrere geographisch verteilte Datenzentren, um steigenden Nutzeranforderungen gerecht zu werden. Eine verteilte Infrastruktur von Datenzentren verändert die Cloud-Kontrollregeln, da Energiekosten von den aktuellen regionalen Elektrizitätspreisen und Temperaturen (sog. geozeitliche Einflussfaktoren) abhängen. Zusätzlich hängt die Preispolitik, über die Cloud-Provider Rechenressourcen anbieten, vom Quality-of-Service (QoS) ab. Auf Grund dieser Preisschemata und der steigenden Energiekosten in Datenzentren, ist das Austarieren zwischen Energieersparnissen und damit einhergehenden Verlusten an Leistung und Gewinnen ein herausforderndes Problem für Cloud-Provider. Bestehende Cloud-Kontrollmethoden existieren nur für einzelne Datenzentren oder berücksichtigen die verfügbaren Cloud-Kontrollaktionen nicht, die zur Reduzierung von Energiekosten in geographisch verteilten Datenzentren führen können.

In dieser Dissertation schlagen wir eine durchdringende Cloud-Kontrollmethode vor, die aus mehreren Methoden für dynamische Resourcereallokation und Hardwarekonfiguration basierend auf volatilen geozeitlichen Einflussfaktoren besteht. Die vorgestellten Methoden betrachten den QoS-Einfluss von Cloud-Kontrollaktionen, sowie Limits für die Datenqualität von Vorhersagemethoden von Zeitreihen. Wir präsentieren einen Cloud-Kontrollmechanismus, der zukünftige Erweiterungen, falls neue Komponenten zur Entscheidungsfindung hinzugefügt werden müssen, ermöglicht. Außerdem stellen wir neuartige Preisschemata vor, die die Verfügbarkeit der Rechenressourcen, sowie die Kosten, die durch unsere Cloud-Kontrollmethode entstehen, in Betracht ziehen, um flexibles, energiebewusstes und hochperformantes Cloud-Computing zu ermöglichen.

Die Cloud-Kontrollmethoden werden empirisch und durch Simulationen evaluiert, die u.a. auf historischen Daten von Elektrizitätspreisen, Temperaturen und Auslastung beruhen. Die potentiellen Energiekostenersparnisse werden geschätzt, indem unsere Methoden mit Methoden, die Stand der Technik sind, verglichen werden. Wir untersuchen eine Vielzahl an Einflussparametern,

um eine Reihe von Empfehlungen für die praktische Verwendung unserer Methoden in Cloud-Systemen abzugeben. Die Resultate zeigen, dass signifikante Energiekostenersparnisse möglich sind, ohne das QoS oder die Gewinne von Services im geographisch verteilten Cloud-Computing zu beeinträchtigen.

Contents

List of Figures

List of Tables

List of Algorithms

Previous Publications

This thesis is based on work published in peer-reviewed scientific journals, conferences and workshops. Here we list the six core papers used as the foundation of this thesis. Parts of these papers are contained in verbatim without explicitly being referenced. Additionally, we also list other publications in the domain of cloud computing and other fields of computer science that were not included in this thesis, but where the author of this thesis is a co-author.

Refereed Publications in Journals

1. **Dražen Lučanin**, Ivona Brandić. *Pervasive Cloud Controller for Geotemporal Inputs.* IEEE Transactions on Cloud Computing, 2016. doi:10.1109/TCC.2015.2464794

2. **Dražen Lučanin***, Ilia Pietri*, Simon Holmbacka*, Ivona Brandic, Johan Lilius, Rizos Sakellariou. *Performance-Based Pricing in Multi-Core Geo-Distributed Cloud Computing.* IEEE Transactions on Cloud Computing (under review, *equal contribution).

Refereed Publications in Conference Proceedings

3. **Dražen Lučanin**, Ilia Pietri, Ivona Brandić, Rizos Sakellariou. *A Cloud Controller for Performance-Based Pricing.* 8th IEEE International Conference on Cloud Computing (CLOUD 2015), 27 June – 2 July, 2015, New York, USA. doi:10.1109/CLOUD.2015.30

4. **Dražen Lučanin**, Foued Jrad, Ivona Brandić, and Achim Streit. *Energy-Aware Cloud Management through Progressive SLA Specification.* 11th International Conference on Economics of Grids, Clouds, Systems, and Services (GECON 2014). 16–18 September, 2014, Cardiff, UK. doi:10.1007/978-3-319-14609-6_6

5. **Dražen Lučanin**, Ivona Brandić. *Take a break: cloud scheduling optimized for real-time electricity pricing.* Proceedings of the 3rd International Conference on Cloud and Green Computing (CGC), 30 September – 2 October, 2013, Karlsruhe, Germany. doi:10.1109/CGC.2013.25

Refereed Publications in Workshop Proceedings

6. **Dražen Lučanin**, Michael Maurer, Toni Mastelić, Ivona Brandić. *Energy Efficient Service Delivery in Clouds in Compliance with the Kyoto Protocol.* 1st International Workshop

on Energy-Efficient Data Centers, 8th May, 2012, Madrid, Spain. doi:10.1007/978-3-642-33645-4_9

Other Refereed Publications

7. Soodeh Farokhi, Pooyan Jamshidi, **Dražen Lučanin**, Ivona Brandić. *Performance-based Vertical Memory Elasticity.* 12th IEEE International Conference on Autonomic Computing (ICAC 2015), 7–10 July, 2015, Gronoble, France. doi:10.1109/ICAC.2015.51

8. Toni Mastelić, **Dražen Lučanin**, Andreas Ipp, Ivona Brandić. *Methodology for trade-off analysis when moving scientific applications to the Cloud.* CloudCom 2012, 4th IEEE International Conference on Cloud Computing Technology and Science. 3–6 December, 2012, Tapei, Taiwan. doi:10.1109/CloudCom.2012.6427575

9. Dragan Gamberer, **Dražen Lučanin**, Tomislav Šmuc. *Descriptive modeling of systemic banking crises.* The 15th International Conference on Discovery Science (DS 2012), 29–31 October, 2012, Lyon, France. doi:10.1007/978-3-642-33492-4_8

10. Matija Gulić, **Dražen Lučanin**, Nina Skorin-Kapov. *A Two-Phase Vehicle based Decomposition Algorithm for Large-Scale Capacitated Vehicle Routing with Time Windows.* Proceedings of the 35th International Convention on Information and Communication Technology, Electronics and Microelectronics – MIPRO, 21–25 May, 2012, Opatija, Croatia.

11. **Dražen Lučanin**, Ivan Fabek, Domagoj Jakobović. *A visual programming language for drawing and executing flowcharts.* Proceedings of the 34th International Convention on Information and Communication Technology, Electronics and Microelectronics – MIPRO. 23–27 May, 2011, Opatija, Croatia. **Best student paper award.**

12. Matija Gulić, **Dražen Lučanin**, Ante Šimić, Šandor Dembitz. *A digit and spelling speech recognition system for the Croatian language.* Proceedings of the 34th International Convention on Information and Communication Technology, Electronics and Microelectronics – MIPRO. 23–27 May, 2011, Opatija, Croatia.

CHAPTER 1

Introduction

Cloud computing is becoming the prevalent way to deliver modern software. The formal definition of cloud computing was given in [81] as *a large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet.* In essence, computational resources are provided to users remotely over the Internet.

The worldwide public cloud services market is projected to reach $204 billion according to a 2016 Gartner report [25]. A reason for the success of cloud computing is that it enables users to get access to computational resources in a black box manner, without necessarily having expert knowledge required to set up and maintain all the underlying technology. They can instead focus on their own domain of work. This separation of concerns fuels the modern information technology (IT) sector, enabling numerous research institutions, existing companies as well as emerging startups to innovate and address problems in all areas of human work.

In this thesis we focus on infrastructure as a service (IaaS) and public clouds. It is the fastest growing segment of the cloud services market for several years already [25, 14]. Examples of public IaaS clouds are the Amazon Elastic Compute Cloud (Amazon EC2) [21] and the Google Compute Engine [26]. Public clouds are a popular cloud computing model where access to computational resources is provided to customers as a business service on the open market. Parts of the described work can be applied to other models as well, however. For example, the platform as a service (PaaS) model also relies on an underlying infrastructure layer, large private clouds might also require energy efficient management similar to public clouds etc. We approach our research from the perspective of a cloud provider, presenting methods mainly meant for cloud providers to implement.

To describe the IaaS public cloud model, it is useful to look at the individual components necessary for such a service. The service provider, i.e. the cloud provider owns a number of physical machines located in one or more data centers (warehouse-like buildings equipped for housing, powering and cooling physical computer infrastructure). Using virtualisation technology, each physical machine (PM) can host multiple virtual machines, isolated from each other and

having access to a certain amount of physical resources (CPU cores, RAM, storage, network bandwidth). The virtual machine (VM), an IaaS product, is offered as a computational resource to customers for usage over the Internet. The VM's usage is monitored to calculate the total price charged to the user for the service. The VM pricing, e.g. hourly price, and QoS, e.g. the VM's technical specifications and availability, are defined in a service level agreement (SLA), a contract between the cloud provider and the customer. We define cloud control as methods for choosing the actions a cloud provider can apply to VMs and underlying PMs that affect computational resource usage and the QoS.

The high level goal of the research described in this thesis is to explore and evaluate new cloud control and VM pricing techniques that improve energy efficiency in IaaS public clouds. More specifically, the focus is on the challenges present in cloud systems consisting of multiple geographically distributed data centers where environmental conditions such as electricity prices and cooling efficiency change over time between data center locations and thus require new cloud control and SLA specification approaches.

To introduce the work presented in this thesis, in the following section we describe the problems motivating our research. In Section 1.2 we list the individual research questions we address. In Section 1.3 we summarily present the contributions of our thesis corresponding to the research questions. In Section 1.4 we give a general methodological framework used throughout the thesis. Section 1.5 presents the organisation of the remainder of the thesis.

1.1 Problem Statement and Research Goals

To satisfy growing cloud computing demands, data centers are consuming more and more energy, accounting for 1.5% of global electricity usage [103] and annual electricity bills of over $40M for large cloud providers [155]. In fact, the ICT sector's 2% global CO_2 emissions have surpassed those of aviation [1], making energy efficiency of data centers a major environmental issue. At the same time, a trend of more geographically distributed data centers can also be seen, e.g. Google has twelve data centers across four continents. As new paradigms develop, such as smart buildings with integrated data centers [153], computation is more and more shaping as a distributed utility. Such cloud deployments result in dynamically changing energy cost conditions and require new approaches to cloud control.

Assorted technological innovations have brought forth the optimisation of several independent systems that affect cloud operation, creating a heterogeneous and dynamically variable environment. The technologies of the next-generation electricity grid known as the "smart grid", distributed power generation, microgrids and deregulated electricity markets have lead to demand response and real-time electricity pricing (RTEP) options where electricity prices change hourly or even by the minute [181, 178]. Additionally, new solutions for cooling data centers (an energy overhead reported to range from 15% to 45% of a data center's power consumption [39]) based on outside air economizer technology result in cooling efficiency depending on local weather conditions. We call such time- and location-dependent factors *geotemporal inputs*. Geotemporal inputs may also include renewable energy availability [86], peak load electricity pricing [116] or demand response [124, 47] as they constitute time- and location-dependent factors that impact the final energy costs as well. To reduce energy costs, new energy-aware cloud control methods

are needed that can leverage geographical data center distribution together with geotemoral inputs such as electricity prices [178] and cooling efficiency [186].

Furthermore, as IT-based optimisation solutions enter more and more domains, we may expect the emergence of new geotemporal inputs in the future. Examples include more options for precisely calibrating electricity usage and pricing in smart grids, local renewable energy generation, further geographical distribution and bringing data centers closer to users through smart buildings [153]. As a result, more advanced metering infrastructure for quantifying cloud service demand and usage through smart cities, smart homes, mobile technology or more generally the Internet of Things (IoT) will capture more data, allowing cloud providers to make better informed decisions. Hence, to account for cloud environment evolution, it has to be possible to extend the cloud controller with yet-to-be-realised geotemporal inputs and other factors.

A side-effect of energy-aware cloud control is that it may impact the QoS. For example, VM availability can be reduced, because certain management actions like VM migrations cause temporary VM downtimes [121]. As long as the resulting availability is higher than the value guaranteed in the SLA, cloud providers can benefit from the cost savings. However, current cloud providers only offer high availability SLAs, e.g. 99.95% in case of Google [26] and Amazon [21]. Such SLAs do not leave enough flexibility to apply energy-aware cloud management or result in SLA violations. Therefore, new SLA specification models are needed that facilitate energy-aware cloud control.

As IaaS clouds are the fastest growing segment of the cloud services market [25], their business models are continuously evolving. New *performance-based pricing* pricing models are being introduced by cloud providers such as ElasticHosts [18] and CloudSigma [17] that radically change the cloud computing revenue models and require new cloud control approaches. In performance-based pricing, the cost of VM provisioning is based on the selected CPU frequency along with the allocated amount of RAM and the use of other resources. The VM's performance can be modified by choosing from a range of CPU frequencies and matching prices – even at runtime. A power management action commonly used to reduce energy costs is CPU frequency scaling [142]. Potential energy savings can be estimated based on geotemporal inputs. CPU frequency scaling actions, however, also cause service revenue losses in performance-based pricing. Hence, cloud control approaches that balance energy savings and service revenue losses caused by CPU frequency scaling in performance-based pricing are needed.

1.2 Research Questions

After giving the detailed problem description in the previous section as a context for the issues that motivate our research, we describe the open research questions that we addressed in this thesis. The first two research questions focus on energy efficiency in cloud control, while the next two consider the effects of energy-aware cloud control on QoS, SLA specification and pricing.

RESEARCH QUESTION 1. *How can we empirically reduce energy consumption in IaaS clouds based on real-time electricity prices?*

Existing VM scheduling methods that can be applied empirically in IaaS clouds typically monitor resource usage and performance, adhering to the agreed constraints and improving

energy efficiency in a best-effort manner by throttling any surplus resources [138, 44]. What such methods usually neglect is the complexity of the electrical grid that powers data centers and that not all energy has the same environmental impact – it differs between generators, depends on demand, grid congestion, time of day, weather conditions and many other factors. An important aspect of current electrical grids is that due to huge demand peaks during midday, electricity often gets generated in more environmentally unsustainable ways such as by diesel generators [36], which is reflected in higher real-time electricity prices [108]. An open problem, therefore, is to integrate such real-time factors into existing cloud systems, such as the open source OpenStack cloud manager, distribute computational resource usage to reduce energy costs and measure the potential savings.

RESEARCH QUESTION 2. *How can we dynamically control cloud resources based on multiple geotemporal inputs to improve energy efficiency and preserve QoS?*

With multiple geotemporal inputs and QoS constraints, new challenges emerge that existing cloud control approaches do not wholly address. Initial VM placement based on geotemporal inputs researched in grid computing [180] or network request routing [155] never reallocates a running VM. Dynamic VM consolidation methods using live VM migrations [79, 43, 138] are focused on a model suitable for a single data center, where no cost heterogeneity inherent to geotemporal inputs is considered. We explore methods that address all these issues, so-called pervasive control, i.e. controlling the cloud's resource allocation dynamically to both consolidate resources and adapt to multiple geotemporal inputs by utilising cost-efficient data centers through long-term planning facilitated by time series forecasting. A challenge in pervasive cloud control is the VM migration action overhead of the VM downtime during the transfer [121]. Forecasting of geotemporal inputs is necessary to find the optimal balance between energy cost saving and VM migration overhead trade-offs that minimise QoS degradation. With time series forecasting that enables long-term planning, the issue of data quality also has to be considered to account for the forecasting accuracy and reach. Additionally, designing a controller to enable adding new geotemporal inputs is necessary to integrate the solution into diverse cloud deployments.

RESEARCH QUESTION 3. *How can we define SLAs for energy-aware cloud services adapted to geotemporal inputs that affect the QoS?*

The challenges of pervasive cloud control and VM scheduling according to volatile geotemporal inputs are that too frequent VM pausing or migrations to reallocate the cloud's resource consumption cause downtimes [121] which harm the QoS and breach VM availability terms specified in the SLA. Various SLA approaches alternative to the typical fixed-price VMs exist, such as auction-based price negotiation in Amazon spot instances [61] or calculating costs per resource utilisation [48]. However, the state of the art is still offering only high availability VMs, without enough flexibility to apply energy-aware cloud control. Hence, estimating availability and price values that can be guaranteed in SLAs for VMs managed based on geotemporal inputs is still an open research issue. This problem is challenging, because exact VM availability and energy costs depend on electricity markets, weather conditions, application memory access patterns and other volatile factors. Methods for analysing the effects of cloud controllers to determine

the availability and price terms that can be specified in SLAs are necessary to support both energy-aware cloud control and a predictable QoS.

RESEARCH QUESTION 4. *How can we adapt cloud control to performance-based pricing, modern CPU architectures and variable application workloads?*

Another approach to reduce energy consumption in data centers is by applying CPU frequency scaling, however this reduces service revenue under the emerging performance-based VM pricing schemes. Cloud control solutions relying on CPU frequency scaling exist, e.g. [175, 167], but only consider fixed VM pricing where the trade-offs of energy savings and performance-based VM pricing are not considered. The currently existing clock frequency governors available at the operating system level that adjust CPU clock frequency according to workload changes have proven to be inefficient in responding to the required VM performance level [91, 92]. Other open challenges are that modern physical machines have multiple CPU cores with complex utilisation-to-power-dissipation models. Additionally, besides the traditional CPU architecture like Intel's, smartphone-technology-based ARM CPU architectures are emerging in large scale cloud platforms [157, 82] with significantly different power models. Finally, the performance impact of the clock frequency may also vary between different workloads [168]. For example, CPU-bound workloads are more sensitive to the reduction in frequency, while the performance of I/O-bound workloads is less affected. The sensitivity of the workload to CPU frequency reduction is called the workload's CPU-boundedness β following the approach in [76]. A cloud control solution has to model and consider an environment with performance-based pricing, multi-core CPUs, different CPU architectures and variable workloads to be of practical relevance.

1.3 Scientific Contributions

Based on the research questions identified in the previous section, we now list the individual contributions presented in this thesis. Each of the contributions was previously published as a peer-reviewed paper that is referenced with the corresponding contribution in this section. A diagram showing the relation between the different research questions and scientific contributions in a geographically distributed cloud system is shown in Fig 1.1.

SCIENTIFIC CONTRIBUTION 1. *Experimental evaluation of a VM scheduling method for reducing energy costs under real-time electricity pricing.*

We introduce the grid-conscious cloud model which relies on computation elasticity to optimise energy usage under dynamic electricity prices. Building on this principle, to validate its feasibility in real-life cloud systems, we present a scheduling method that analyses historical electricity price data [22] to determine the most probable peak hours and then pauses the managed VMs. We implemented this scheduling method – the peak pauser cloud controller on the industry-standard OpenStack cloud manager [105] to empirically measure its effectiveness. We calculated further savings projections based on the available assessments of hardware used in Google's production environment [77] and obtained encouraging results. Since pausing VMs is an invasive action towards the user, we present *green instances*, a business option similar to Amazon's

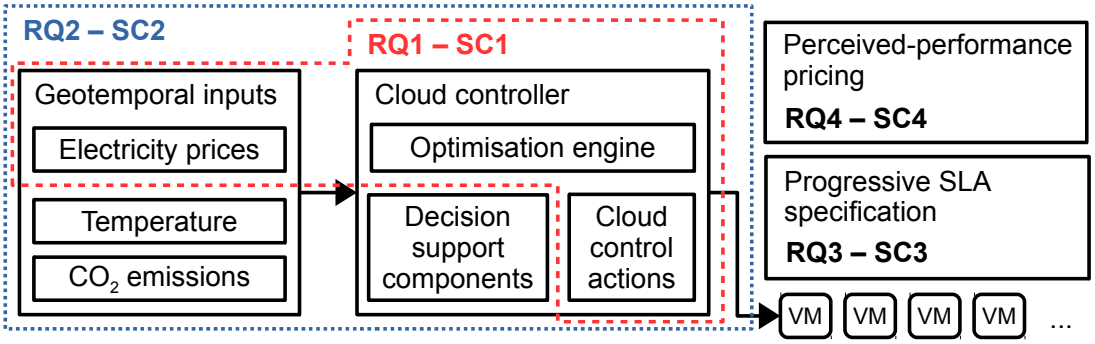


Figure 1.1: Organisation of scientific contributions.

spot instances [29] where the user is offered a better service price in exchange for reduced VM availability. This contribution addresses research question (RQ) 1. It was previously published in [132] and is presented in Chapter 4 of this thesis.

SCIENTIFIC CONTRIBUTION 2. *A pervasive cloud controller considering multiple geotemporal inputs that unifies SLA compliance, VM and PM per-resource constraint checking, live VM migration overhead, as well as data center energy costs.*

In this thesis we propose a novel pervasive cloud controller designed for resource allocation optimisation that efficiently utilises cloud infrastructure, accounting for geographical data center distribution under geotemporal inputs. Our optimisation engine supports components for costs based on geotemporal inputs, VM migration overheads, QoS requirements and other inputs to be composed in a unified optimisation problem specification. A schedule of VM migrations is planned ahead of time in a forecast window. This allows the controller to minimise energy costs by planning over a long-term period such as hours or days, while retaining the required QoS, i.e. not incurring too frequent VM migrations. To assess the application of our pervasive cloud controller in diverse cloud deployments, we present a number of guidelines showing how the effectiveness changes under different geotemporal input patterns, geographical distributions and forecast data quality.

As a proof of concept evaluation we present an implementation of the pervasive cloud controller based on a hybrid genetic algorithm for optimising the schedule of VM migrations. A time-series-based schedule representation is developed for integration with geotemporal inputs to facilitate long-term planning. A realistic duration-agnostic model, with no a priori VM lease duration knowledge assumption, improves the compatibility with real cloud deployments, such as Amazon EC2, Google Compute Engine or private OpenStack clouds. Multiple decision support components including energy cost, QoS, migration overhead and capacity constraints are combined into an extensible fitness function.

We evaluate the pervasive cloud controller in a large-scale simulation consisting of 10k VMs using historical electricity price and temperature traces to show the resulting energy cost savings and QoS impact. Based on our simulation results, energy cost savings can be increased

up to 28.6% compared to a baseline scheduling algorithm [45] with dynamic VM consolidation. Furthermore, we expand the evaluation to provide guidelines for cloud providers in terms of how different geotemporal input value ranges and geographical data center distributions affect the method's effectiveness. We provide a data quality analysis by evaluating the controller under different forecasting errors. Finally, we validate the architecture's extensibility by performing the simulation with different subsets of decision support components. This contribution addresses RQ 2. It was previously published in [126] and is presented in Chapter 5 of this thesis.

SCIENTIFIC CONTRIBUTION 3. *A progressive SLA specification based on VM availability and cost analysis from historical cloud control actions based on geotemporal inputs.*

In this thesis, we propose a novel approach for estimating the optimal number of SLAs, as well as their availability and price values under energy-aware cloud management. Specifically, we present a method to analyse past traces of dynamic cloud management actions based on geotemporal inputs to estimate VM availability and price values that can be guaranteed in an SLA. Furthermore, we propose a progressive SLA specification where a VM can belong to one of multiple treatment categories, where a treatment category defines the type of energy-aware management actions that can be applied. An SLA is generated for each treatment category using our availability and price estimation method.

We evaluate our method by estimating availability and price values for SLAs of VMs managed by two energy-aware cloud controllers – the pervasive cloud controller utilising live VM migration from scientific contribution (SC) 2 and the peak pauser cloud controller from SC 1. We evaluate the SLA specification in a user SLA selection simulation based on multi-auction theory [104] using Wikipedia and Grid'5000 workloads to represent multiple user types. Our results show that more users with different requirements and payment willingness can find a matching SLA using our specification, compared to existing high availability SLAs. Average energy savings of 39% per VM can be achieved due to the extra flexibility of lower availability SLAs. Furthermore, we determine the optimal number of offered SLAs based on customer conversion. This contribution addresses RQ 3. It was previously published in [73] and is detailed in Chapter 6.

SCIENTIFIC CONTRIBUTION 4. *A cloud control method for allocating VMs and scaling CPU frequencies with perceived-performance VM pricing and non-linear power modelling.*

In this thesis, we introduce a compound cloud control model which considers multiple factors representative of modern cloud systems. We combine: (1) Realistic power modelling accounting for multi-core, Intel and ARM architectures; (2) Energy cost calculation based on geotemporal inputs; (3) Performance-based VM pricing; (4) Variable VM CPU-boundedness that determines the performance impact of CPU frequency scaling.

To describe real-world power dissipation behaviour, we developed a non-linear power model based on real experiments performed on multi-core Intel and ARM CPU architectures representative of modern data center infrastructures [157, 82]. As we show, on such power models, traditional race-to-idle approaches [162, 164] are no longer valid, which also motivates the cloud control method we introduce.

To tackle varying VM workloads, we propose a novel perceived-performance pricing scheme for determining the VM price based on the application-level performance. This scheme allows

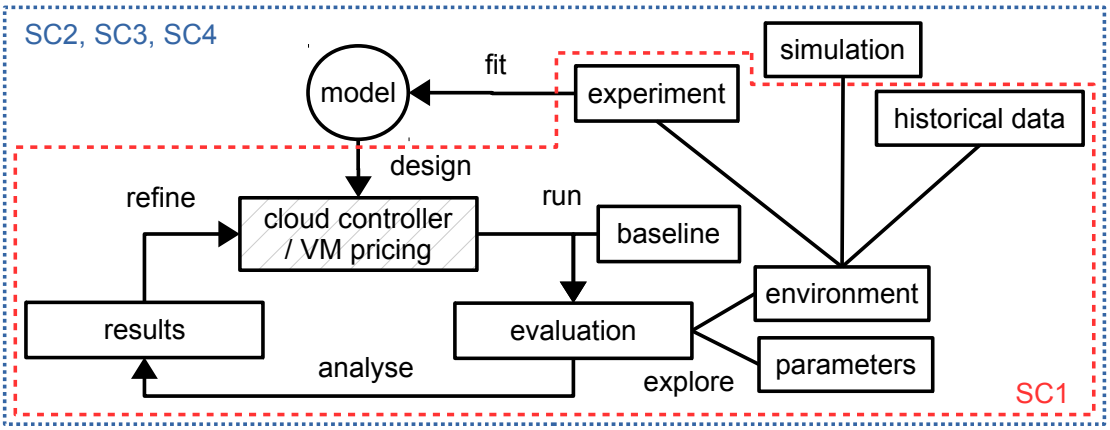


Figure 1.2: Research methodology.

energy-aware cloud control that treats VMs differently based on the actual impact that CPU frequency scaling will have on their workload performance, considering their measured CPU-boundedness.

To address the data center energy consumption and performance-based VM revenue trade-offs, we introduce the Best Cost Fit Frequency Scaling (BCFFS) cloud controller. The controller we propose uses our multi-core power model for ARM and Intel CPU architectures in an energy calculation method that factors in geotemporal inputs from multiple geo-distributed data centers. To account for performance-based VM pricing, ElasticHosts [18] and CloudSigma [17] price data was used to model their behaviour and precisely compute the effects of each CPU frequency level. The BCFFS cloud controller then combines both models in a two-phase algorithm, where firstly VMs are allocated between geo-distributed data centers and subsequently CPU frequencies are set for each PM where energy savings exceed service revenue losses.

The controller and the models were mapped onto the Philharmonic simulator [129]. Simulations with a wide range of scenarios are used to estimate the energy savings and service revenue stemming from our cloud control approach. The results obtained by the BCFFS cloud controller are compared and evaluated using two baseline controllers [44] and historical traces of real-time electricity prices [33] and temperatures [129]. The VM CPU-boundedness values used in the simulation are distributed according to the PlanetLab [12] dataset of VM CPU usage. The results indicate that energy savings up to 14.57% without significant service revenue reductions can be achieved using the BCFFS cloud controller. This scientific contribution addresses RQ 4. It includes combined work published in [127] and [128]. The details are presented in Chapter 7.

1.4 Methodology

After an overview of the thesis contributions, in this section we describe the general methodology applied in each of the contributions to evaluate their effectiveness towards addressing the corresponding research questions. Generally speaking, our work combines aspects of several

paradigms: (1) empirical research where mathematical models are obtained from real-world data collected in practical experiments; (2) software simulation, an approach often used in computer science, where real-world behaviour is approximately imitated (i.e. simulated) in a computer program developed for this purpose; (3) data analysis or data science, where statistical methods are applied to process the data collected from experiments or simulations and extract conclusions.

To analyse the results, we relied on the data analysis tools and libraries available in the Python programming language ecosystem. Most importantly, we used the Jupyter (formerly IPython) interactive notebook [149], the Pandas [139] library for tabular data manipulation and time series analysis and the Matplotlib [97] plotting library to generate most of the figures in this thesis.

The general applied methodological approach is illustrated in Fig. 1.2. The steps are:

1. Model the properties of cloud systems and the effects various cloud control various actions have on them. The models are obtained by performing real-world experiments, making a hypothesis of the mathematical form that could describe the behaviour and fitting the model's parameters using methods such as linear regression.

2. Design a cloud control algorithm based on the modelled cloud properties and actions to improve the output metrics, such as energy consumption, cost and QoS. Alternatively, in some of the contributions, primarily a VM pricing scheme is designed.

3. Define an environment for evaluating the cloud controller as a real-world experiment or as a simulation resembling real-world behaviour using the available cloud model and historical data traces.

4. Use said evaluation environment to evaluate the cloud controller and possible existing cloud controllers as a baseline to compare the results. Based on the results, refine the design of the controller and repeat the process. Explore different controller and input data parameters in the process as well.

The methodology outlined above with subtle variations is applied to all our scientific contributions. We will shortly explain how each of the scientific contributions adheres to this methodology, as we also illustrate in Fig. 1.2.

In SC 1 we evaluate the peak pauser cloud controller compared to applying no actions as a baseline in an experimental deployment on the OpenStack cloud manager [105]. We analyse results collected from a wattmeter to calculate the energy costs for both scenario based on historical real-time electricity price data and compare the energy and cost savings.

SC 2 includes several models fitted from experimental data such as the VM migration energy overhead and downtime, the cooling model and add our own model for combining the cost components. We use the Philharmonic simulator [72] we developed to evaluate the proof-of-concept pervasive cloud controller and compare it to two baseline methods [44]. We analyse the effects of an assortment of parameters, e.g. the effects of different time series forecasting errors. Historical electricity price [33] and temperature [19] datasets for US-only and worldwide data center locations were used in the simulations.

SC 3 evaluates a progressive SLA specification approach using a dataset of historical energy-aware cloud control traces (obtained in turn using past geotemporal inputs). The SLA specification

method was evaluated in a user behaviour simulation based on multi-auction theory [104] where the users' requirements were modelled from historical Wikipedia and Grid'5000 workloads.

Finally, SC 4 evaluates a cloud controller based on VM placement and CPU frequency scaling, using a methodology similar to SC 2 using the Philharmonic simulator and historical geotemporal inputs. Additional modelling was applied to make the simulations more practically relevant, including workloads with different CPU boundedness properties, a power models for multi-core CPUs and both Intel and ARM architectures. A variety of parameters were analysed in the results, such as how different pricing schemes, cloud providers or CPU architectures affect the potential energy savings.

1.5 Thesis Organisation

The remainder of the thesis is organised as follows:

- Chapter 2 systematically examines the various research branches grouping work related to ours. We cover the state-of-the-art methods in each of these branches and highlight some of the open challenges that we address in our work.
- Chapter 3 gives some more background on geotemporal inputs and methods to forecast their future behaviour. We also enter into the details of CO_2 equivalent (CO_2e) trading where CO_2e emissions become a cost component similar to electricity prices, a model we previously published in [130].
- Chapter 4 offers an empirical approach to designing a cloud control method for reducing energy costs in the context of real-time electricity pricing. The focus is mostly set on integration with real-life systems such as OpenStack and measuring the outcome using wattmeters to validate the method's feasibility.
- Chapter 5 presents a pervasive cloud controller that accounts for multiple geotemporal inputs, including real-time electricity pricing and temperature-dependent cooling efficiency and defines a more general approach for managing an extensible set of geotemporal inputs and decision criteria. We propose a proof-of-concept genetic algorithm for controlling VM migration in a geographically distributed cloud and show substantial energy cost savings possible using the method.
- Chapter 6 focuses on the QoS effects energy-aware methods from the previous chapters pose. We present a method for progressive SLA specification where VMs are priced and controlled differently based on the selected SLA. A customer behaviour simulation methodology is presented to evaluate the economical outcome of this progressive SLA system.
- Chapter 7 analyses the emergence of performance-based pricing schemes in IaaS markets where VM properties such as CPU frequency can be changed at runtime and affect the hourly VM price. We present a detailed multi-core power consumption model based on different CPU frequencies. We then propose a novel perceived-performance pricing

model which also considers the CPU-boundedness properties of the VM workload. Finally, we propose a cloud control method that balances the revenue and energy cost impact to determine optimal CPU frequency scaling actions across a geographically distributed cloud.

- Chapter 8 summarises the thesis, gives the concluding remarks and outlines limitations and possible future work directions.

CHAPTER 2

Related Work

In the domain of energy efficiency in cloud computing, there is a large body of related work that has been published. We structure related work into the following list of categories: (1) cloud control methods – cloud resource allocation techniques and cloud management actions which aim to improve energy efficiency in cloud computing; (2) energy efficiency legislation – existing rules, regulations and best practice recommendations from optimising the whole data center efficiency towards optimising its constituting parts; (3) SLA management methods – existing and emerging VM pricing models suited for energy-aware cloud management; (4) Power modelling – methods to mathematically describe the power consumption behaviour of cloud systems and cloud control actions that can be applied to it. We will examine each of these two groups separately now.

2.1 Cloud Control

Energy efficiency is often considered in cloud control – a survey is available in [46]). We categorises cloud control approaches relevant to the work presented in this thesis into methods for: (1) scaling CPU frequencies in PMs hosting VMs, (2) initial VM placement, (3) subsequent dynamic VM reallocation using live migration and (4) the optimisation theory used to find the best allocation strategy.

2.1.1 CPU Frequency Scaling

Frequency scaling is the focus in many studies with the aim to reduce power consumption by decreasing the CPU frequency [76, 175, 179]. The cloud scheduler in [179] sorts and allocates the incoming jobs to VMs based on the user SLAs. The minimum resource requirements are allocated to each VM and the CPU frequencies of the PMs with low load are reduced so that resource wastage is minimised without affecting the performance of the executing jobs. In [175], the proposed scheduler allocates the queued VMs to PMs, while reducing the CPU frequencies at runtime so that VM performance requirements can be met, preferring PMs that operate at lower frequencies. As opposed to related work, the controller we propose in Chapter 7 scales

the CPU frequencies taking into account the workload CPU-boundedness while controlling the impact of frequency reduction on the provider's profit. The impact of frequency scaling on workload performance has been investigated in many studies [94, 76, 142, 83]. The authors in [94] introduce a metric to model workload CPU-boundedness based on the observation that frequency scaling potentials depend on the workload CPU-boundedness, with less CPU-bound applications being benefited more. In [142], the authors investigate the power-performance characteristics of systems with frequency scaling capabilities and introduce a metric to determine energy-efficient performance points to operate the system. This is also the focus in [83], investigating the impact of frequency scaling on workload performance for different HPC workloads in order to achieve energy-performance trade-offs.

Although dynamic voltage & frequency scaling (DVFS) cloud controllers have been proposed before [95, 169, 101, 34], our adaptive approach scales the operating frequencies based on the VM CPU-boundedness and the impact frequency reduction has on the provider's gross profit under performance-based pricing. Also, in most papers multi-core modelling is not considered or is simplified as a linear combination of the number of cores used. The clock frequency of the systems used in cloud platforms are usually modelled according to its dynamic power as a product of the clock frequency and the core voltage. In contrast to such systems, we focus on adopting a more accurate multi-core power model; still simple enough to be integrated in real systems. The model accounts for real-world influences more accurately such as the heat dissipation influencing the static power significantly [99].

Besides for dynamically scaling CPU, other hardware-related approaches include completely suspending PMs [69], which we analysed in the context of geotemporal inputs in Chapter 4. Insights about choosing hardware based on power consumption are given in [44, 42].

2.1.2 Initial VM Placement

Looking at the group of methods that only perform initial placement and consider geotemporal inputs during host selection, the approach was pioneered by Quereshi et al. [155]. Their work shows the potential of optimising distributed systems (adaptive network routing in content delivery networks) for RTEP, estimating savings up to 40% of the full electricity cost. Similar routing approaches are explored in [158, 120, 117] and considering both electricity prices and CO_2 emissions in [71]. Initial placement is also researched in the context of map-reduce jobs [56] and based both on RTEP and cooling in computational grids in [89, 123]. Load balancing where RTEP are considered is studied in [131]. Both web request routing and map-reduce jobs are different than the IaaS VM hosting model we are researching in that they require only initial placement. Methods presented in [89, 123] take into account cooling and RTEP to place jobs in a grid system, but apply them on a grid system where jobs need to be scheduled only initially with no reallocation. A theoretical analysis of placing grid jobs with regards to electricity prices, job queue lengths and server availability is given in [160], but without considering a realistic model of VM migration overhead and QoS impact. The research was extended with a cooling model in [152]. A power-aware job scheduler with no rescheduling and assuming a priori job duration knowledge is presented in [181]. Job scheduling that considers an application's data access requirements, network capacity limits etc. in geo-distributed data centers is presented in [182]. In [66] the HGreen heuristic is proposed to schedule batch jobs on the greenest resource

first, based on prior energy efficiency benchmarking of all the nodes, but not how to optimize a job once it is allocated to a node - how much of its resources is it allowed to consume. A similar multiple-node-oriented scheduling algorithm is presented in [176]. A survey of VM placement methods regarding energy efficiency, SLAs andQoS is given in [151]. Multi-objective approaches [78, 183] are also becoming popular to initial allocation, where it is seen that the scheduler's goal is to satisfy economic cost, energy consumption as well as reliability constraints, which is nearer to our approach of weighing service availability and energy cost and efficiency.

A job scheduling algorithm for geographically distributed data centers with temperature-dependent cooling efficiency is given in [180], but only for initial job allocation and no subsequent migrations. A method for using migrations across geographically distributed data centers based on cooling efficiency is shown in [116], but for a grid system, where job durations are declared by users in advance, unlike cloud systems where VM deletion is entirely up to the user, in a pay-as-you go scheme. Also, no QoS aspects are considered in the optimisation, which can lead to too many migrations per VM.

2.1.3 Dynamic VM Consolidation

The dynamic VM consolidation group includes methods targeted at modern IaaS clouds where live VM migration is used to dynamically reallocate resources and reduce energy consumption. The topic of using live VM migrations to reduce energy consumption in cloud computing has been widely researched [79, 43, 138, 42, 65, 59, 119, 43]. All the methods we found, however, value energy the same, no matter the time or location, and therefore overlook the additional challenges and optimisation potential of geotemporal inputs. Feller et al. proposed a distributed scheduling algorithm for dynamic VM consolidation using live migrations, based on hierarchical group management [79]. Beloglazov et al. [43] introduced a VM consolidation method that minimises the migration frequency in an online controller, taking future workload predictions into consideration. A rule-based VM consolidation approach is developed in [138]. Practical cloud control utilising VM migrations with a focus on high scalability in production VMware systems is researched in [106]. A VM allocation method for gaming clouds that considers QoS and revenue aspects is presented in [93]. Deshpande et al. [70] present a method for coordinating live VM migration considering network traffic usage to not affect application network requirements. Consolidation based on RAM and CPU usage is researched and evaluated on a real data center in [136]. Aikema et al. [30] show a method of rescheduling low-priority jobs in clusters to improve the data center's carbon footprint, which goes in the direction of our green instances in the context of computational grids. A more adaptive method for detecting changes in QoS requirements and subsequent calibration of virtualised resources is presented in [122]. Low-level VM migration time, energy and bandwidth metrics are studied in [121, 31, 145].

2.1.4 Pervasive Control

In a group we named pervasive cloud, VMs are dynamically migrated to adapt both to user requests and changes in geotemporal inputs that enable energy cost savings, while considering forecast data quality and QoS requirements. As already stated, to the best of our knowledge, no cloud control method solves this problem completely. There have been advances in this direction,

however. There have been initial advances in this direction. Cauwer et al. [58] presented a method of applying time series forecasting of electricity prices to detect how a data center's resource consumption should be controlled in a geographically distributed cloud, but for a simplified model with no concrete actions that should be applied on VMs. Determining exact per-VM actions is a challenging trade-off problem, between closely following volatile geotemporal input changes and minimising the number of migrations to retain high QoS, as we will examine in Chapter 5. Time series forecasting is used in the cloud computing domain in [57] to predict application workload in a software as a service (SaaS) cloud to better utilise computational resources. Abbasi et al. [27] started researching migrating VMs based on RTEP, but for a limited workload distribution scenario where a PM hosts only a single VM. Additionally, temperature-dependent cooling energy overhead and forecasting errors are not considered. Our work addresses these challenges through a holistic model supporting multiple decision support components and long-term planning facilitated by forecasting and data quality assertion.

2.1.5 Optimisation Algorithms

Finally, we look at work related to ours from an algorithmic perspective. Approaches for schedule optimisation based on a forecast horizon are explored as rolling-horizon real-time task scheduling in [187] and a genetic algorithm for this purpose was used for multi-airport capacity management with receding horizon control in [96]. A genetic algorithm for optimising energy consumption in computational grids using DVFS is presented in [110]. Ant colony optimisation, also an evolutionary optimisation algorithm is used in [170] to schedule tasks in grid computing. Tabu search, a related meta-heuristic optimisation method, is used for static data center location and capacity planning with a focus on network traffic in [114]. Another multi-objective evolutionary method is biogeography-based optimisation, mimicking the distribution of biological species through time and space, which was used in [185] for VM placement that minimises power consumption, migration time and network traffic. Other approaches use deterministic approaches, such as constrained optimisation problem solving which is used in [63] to determine power saving policies that reduce idle PM power. However, multi-objective evolutionary algorithms are still an active research topic [154, 118, 133], so it is likely that they will find more usage in real-time cloud control scenarios with time.

2.2 Energy Efficiency Legislation

Measures of controlling energy efficiency in data centers do exist – metrics such as power usage efficiency (PUE) [9], carbon usage efficiency (CUE), water usage efficiency (WUE) [8] and others have basically become the industry standards through the joint efforts of policy makers and cloud providers gathered behind The Green Grid consortium [7]. The problem with these metrics, though, is that they only focus on the infrastructure efficiency – turn as much energy as possible into computing inside the IT equipment. Once the power gets to the IT equipment, though, all formal energy efficiency regulation stops, making it more of a black-box approach. For this reason, an attempt is made in our work to bring energy efficiency control to the interior operation of clouds – resource scheduling.

So far, the measurement and control of even such a basic metric as PUE is not mandatory. It is considered a best practice, though, and agencies such as the U.S. Environmental Protection Agency (EPA) encourage data centers to measure it by rewarding the best data centers with the Energy Star award [24].

A good overview of cloud computing and sustainability is given in [85], with explanations of where cloud computing stands in regard to CO_2e emissions. Green policies for scheduling are proposed that, if accepted by the user, could greatly increase the efficiency of cloud computing and reduce CO_2e emissions. Reducing emissions is not treated as a source of profit and a possible way to balance SLA violations, though, but more of a general guideline for running the data center to stay below a certain threshold. We discuss cloud computing in the context of the Kyoto protocol and its cap-and-trade system in Chapter 3.

2.3 SLA Management Methods

The disadvantages of current VM pricing models relying on constant rates have been shown by Berndt et al. [48] using a game theory approach. They propose a method for variable VM pricing, based on the actual VM utilisation. A new charging model for PaaS providers, where variable-time requests can be specified by the users, is developed in [174]. A fine-grained pricing scheme for IaaS providers was proposed in [102] accounting for overhead costs such as the VM startup time. Ibrahim et al. applied machine learning to compensate for interference between VMs in a pay-as-you-go pricing scheme [100]. Though related, Amazon spot instances [61] permanently terminate VMs that get outbidden, hence requiring fault-resilient application architectures. Aside from this, they perform exactly like other Amazon instances, again not allowing temporary downtimes necessary every day for energy-aware cloud management. Alternative auction-based pricing systems are explored in [135, 52]. Amazon spot instances and similar auction-based schemes do show that end users are willing to accept a more complex pricing model to lower their costs for certain applications, indicating the feasibility of such approaches in real cloud deployments. Zhang et al. [184] propose model predictive control as a method for maximizing the provider's revenue by matching customer demand in terms of supply and prices. Toosi et al. [171] present a method for managing a diverse set of pricing options, such as on-demand or spot instances, and solving the optimisation problem of allocating data center capacity to each data plan. Cloud market mechanics are analysed in [53], where service standardisation and automatic adaptation to user requirements using clustering algorithms and monitor is explored and in [54] where a modified cloud market liquidity metric is proposed to monitor market efficiency. A market-based system based on bidding and adapting to user needs is discussed in [60].

But for many of these energy management techniques to have a significant impact, computation has to be flexibly offset, at which point QoS concerns become an issue and it becomes problematic to choose the users most appropriate for such green services. To resolve this issue, the authors in [112] proposed a green cloud architecture, whereby "green broker" middleware manages the selection of the "greenest" cloud provider for users. The green cloud broker depends on a Carbon Emission Directory (CED) that conserves all the data related to energy efficiency of the clouds and a Green Offer Directory (GOD). In contrast, the SLA management approach we present in Chapter 6 relies on QoS requirements and pricing information, letting a service broker

select the best match from a competitive market dominated by energy-conscious clouds. Another expansion of the typical SLA negotiation process is presented in [108], where GreenSLAs are used to define energy-related cloud provider properties. None of the mentioned pricing approaches consider energy-aware cloud management based on geotemporal inputs or the accompanying QoS and energy cost uncertainty, which is the focus of our work.

The basis of our service broker's economic model in Chapter 6 is the utility-based algorithm, which is adopted from multi-attribute auction theory [37], to describe user preferences as a function of weighted SLA attributes taking into account the user's payment willingness. This economic model is well-established in literature in the context of web services for the optimisation of service value networks [113] and for the service portfolio design [109]. It has also been applied in the context of smart grids [80] for the evaluation of variable electricity rates in order to achieve efficient matching of electricity consumer needs and generator capabilities.

2.4 Power Modelling

The literature has shown many power models and approaches to model the power dissipation in computer systems [141, 159, 172, 162, 67, 165]. Most models are constructed bottom-up from physical characteristics on top of which practical aspects such as frequency scaling is applied. The dynamic power dissipation is expressed in many examples [64, 166, 156] as the relation $f \cdot v^{\alpha}$ where f is the clock frequency, v is the core voltage and α is a constant used to comply with the real platform as close as possible.

As the dynamic power dissipation can be expressed with this simple bottom-up formula, the ever growing static power proves more difficult. The leakage current causing the static power is expressed in [107] as a relationship between transistor gate width, thermal voltage and architectural parameters such as the insulation material. Moreover, leakage is also caused by electron tunnelling through the insulator. This means that a bottom-up modelling of static power is significantly more difficult. We instead used a top-down view of the power model, purely based on real-world experiments, which provides a more realistic view of the complete system including cores, buses, memories, temperature, operating system influence and other software.

CHAPTER 3

Background on Geographically Distributed Cloud Computing

As detailed in the previous chapter, cloud computing is revolutionizing the ICT landscape by providing scalable and efficient computational resources on demand. However, the data centers hosting these computational resources are responsible for considerable energy consumption and subsequent CO_2 emissions. In this chapter, we provide background on the energy consumption in the context of geographically distributed clouds. An opportunity to optimise computational resource management lies in the fact that more and more cloud providers own geographically-distributed data centers, with temporal variation of energy and cost efficiency at different locations – different cooling efficiency factors and real-time electricity prices. To tap into such opportunities, there is a need for automatic, fast and precise analysis of dynamically-changing variables, providing insights into their underlying characteristics and future behaviour. Time series analysis and forecasting is a statistical tool that provides such capabilities. In this chapter we give a comparative description and evaluation of two methods for time series forecasting. Automatic Autoregressive Integrated Moving Average Modelling is an expert system that performs adaptive data preprocessing and model fitting resulting in a stationary process representing the series. The Theta method is based on exponential smoothing of the series with random drift. Both methods participated in the M3 competition where they were evaluated on 3003 time series datasets. Additionally, we performed our own evaluation on real-time electricity price and temperature datasets, important in cloud computing. We give an overview of the results of the M3 competition and our simulation in the context of cloud computing.

Another geotemporal input is the CO_2e cost of the energy used by the data centers. It is possible that cloud providers might one day be faced with legislative restrictions, such as the Kyoto protocol, defining CO_2e caps. A lot has been done around energy efficient data centers, yet there is very little work done in defining flexible models considering CO_2. In this chapter we present a first attempt of modelling data centers in compliance with legislation such as the Kyoto protocol. We discuss a novel approach for trading credits for emission reductions across data centers to comply with their constraints. CO_2 caps can be integrated with Service Level

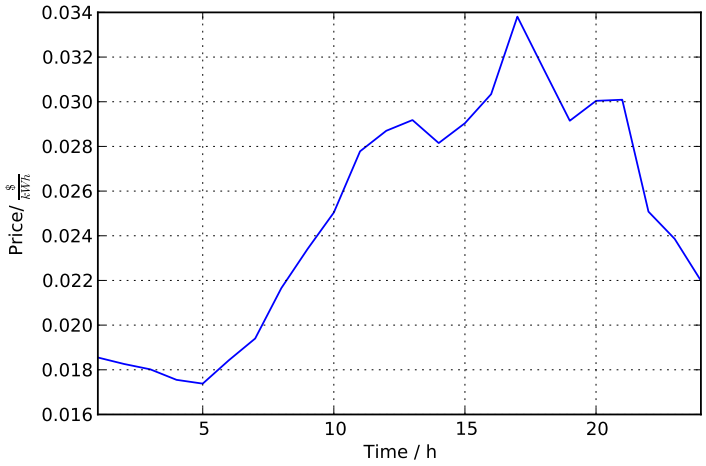


Figure 3.1: A time series of hourly electricity prices in Illinois, USA within a single day (visualisation of the data available in [22]).

Agreements and juxtaposed to other computing commodities (e.g. computational power, storage), setting a foundation for implementing next-generation schedulers and pricing models that support Kyoto-compliant CO_2 trading schemes.

This chapter is structured as follows – forecasting of geotemporal inputs is analysed in Section 3.1. In Section 3.1.1, we first examine each method for time series forecasting individually – Automatic ARIMA Modeling (AAM) and the Theta method.We examine their underlying theoretical mechanisms (autoregressive versus exponentially-weighted moving averages) and the impact they impose. We follow by describing the empirical benchmarks of both methods as part of the M3 competition and in our own simulation where we evaluate them on cloud-related data in Section 3.1.2. We give an overview of the results in Section 3.1.3. In Section 3.2, we detail the CO_2e emission legislation's implications to cloud computing. Section 3.2.1 gives some background as to why cloud computing might become subject to the Kyoto protocol. Section 3.2.2 presents our model in a general CO_2e-trading cloud scenario, we then go on to define a formal model of individual costs to find a theoretical balance and discuss the usefulness of such a model as a scheduling heuristic. Section 3.3 gives a summary of the chapter.

3.1 Forecasting Geotemporal Inputs

To fully optimise energy consumption in cloud computing, it is necessary to consider temporal and geographical variations in electricity prices and colling efficiency. They offer an opportunity to e.g. migrate VMs to regions with optimal dynamic conditions. This way we prioritize VM execution in energy- and cost-efficient space-time windows. Two example geotemporal inputs that affect geographically distributed data centers are:

1. *Real-time electricity prices.* In regions with deregulated electricity markets, electricity is available at prices that change hourly with generation, distribution and demand conditions [178]. We call this purchasing option RTEP. Such markets exhibit a price volatility up to two orders of magnitude higher than that of other commodities [178]. For the market we observed [22], hourly electricity price peaks are more than double the low value within the same day on the average (visible in Fig. 3.1).

2. *Cooling efficiency.* To cool the data center, additional energy is necessary and the exact amount depends on the outside air temperature – cheaper cooling at low temperatures and more expensive cooling at high temperatures [89]. Air temperature also changes dynamically, depending on weather conditions.

To be able to make informed decisions in cloud management based on these dynamic, time-dependent variables, knowledge of their future behaviour is necessary. Time series forecasting, a large sub-field of statistics is concerned with exactly such problems [55]. Time series forecasting is used across many domains such as finance, macro-economy, weather forecasting, signal processing etc. and it is a technique to quickly and automatically predict future variable values. In this section the aim is to give a survey of the state-of-the-art techniques for time series forecasting, their precision and suitability in the context of data important for resource management in cloud computing.

To collect information on the quality of available forecasting methods, the M3 competition [134] was held in 2000. It evaluated the available methods on the same dataset encompassing 3003 time series from different domains. In this section we will examine and compare two interesting methods that took part in this competition. The first method offers a way to automatically build Autoregressive Integrated Moving Average (ARIMA) models and is therefore known as AAM. The AAM method is presented in [143]. It predicts future behaviour of a time series based on the statistical non-stationary data properties. The Theta method presented in [98] is an empirically-successful time series forecasting method (winner of the M3 competition [134]), shown to be equivalent to Simple Exponential Smoothing (SES) – an ad-hoc data smoothing procedure.

After defining the methods, we analyse their effectiveness in the M3 competition, restating some of the findings important for our survey from [134]. The Theta method proved to be the most successful general-purpose forecasting method, while the AAM method achieved better results when there was a longer history available.

3.1.1 Methods for Time Series Forecasting

We define a time series as observations ordered in time:

$$x_t \; ; \; t = 1, \ldots, T, x_t \in \mathbb{R}^n \tag{3.1}$$

For $n = 1$ we have a time seris of scalar values – e.g. hourly observations of the electricity price in Illinois illustrated in Fig. 3.1, while for a larger n we have multi-dimensional data for every point in time. As a simple example of a time series that will be used to illustrate forecasting, daily temperatures in Kraków for four days are shown in Table 3.1.

Date (June, 2013)	18	19	20	21
Temperature (C)	31	32	26	26

Table 3.1: Daily temperatures in Kraków [20]

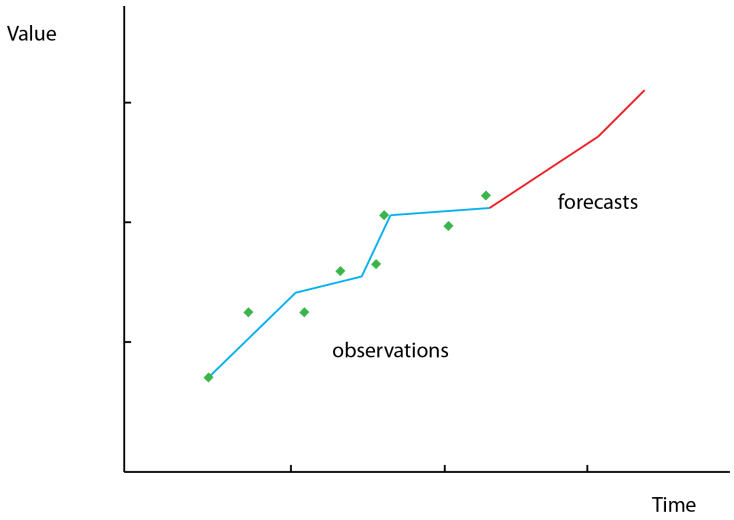


Figure 3.2: An illustration of time series forecasting.

The goal of a time series forecasting method is to take as input a series x_t of time-stamped values over a period in history $t = 1,\dots T$ and produce as output the predicted values for the predetermined number of future time stamps x_t, $t = T+1,\dots T+m$. An illustration of this process is shown in Fig. 3.2. A statistical model (the full line) is built based on known past observations (green dots) to predict future behaviour (the red part of the line). In the remainder of this section, we will show how this is performed by two methods – AAM and the Theta method.

Automatic ARIMA Modeling

An ARIMA process is a statistical model used to describe the behaviour of a time series. It is a generalisation of the Autoregressive Moving Average (ARMA) model. First, an initial step is performed to detect and remove non-stationarity[1] from the data. After this, the time series X_t is represented by an $ARIMA(p',q)$ model:

$$\left(1-\sum_{i=1}^{p'}\alpha_i L^i\right)X_t = \left(1+\sum_{i=1}^{q}\theta_i L^i\right)\varepsilon_t \tag{3.2}$$

where L stands for the lag function, α_i and θ_i parameters have to be fitted on the training data and (p',q) parameters define the order of the autoregressive and moving average parts. Since

[1] Stationary processes are stochastic processes whose joint probability distribution does not change when shifted in time.

fitting an ARIMA process to match the empirical data requires statistical and domain-specific knowledge, Mélard and Pasteels built AAM, an expert system [143] that performs automatic, self-calibrating ARIMA parameter fitting. Some of the features offered by the AAM expert system are:

- Keeping the user informed of the intermediate and final results.
- The model can be modified afterwards.
- Adjusting the level of automation according to the expertise of the user.
- Regarding only the most "substantive models" for lag structures.
- Working efficiently in case of a large number time series dataset.
- Intervention analysis for the appropriate treatment of outliers

This way, complex statistical models of a time series can be built without having to be an expert in the field. The automatic steps applied by the expert system are:

1. Determining the seasonal difference – the Kruskal and Wallis test of seasonal differences is performed, while the autocorrelation function is used to check if the lag coefficient stays normal after the elimination of the seasonal component.
2. Choosing the transformation – the Tau test of rank correlation coefficients is used to find an appropriate transformation of the data to build the ARIMA model.
3. Intervention analysis – a step that automatically handles the outliers in the processed series (e.g. removes them).

The resulting model is checked by analysing the convergence of the optimisation process and afterwards for stationarity and invertibility of the autoregressive and moving average polynomials.

The Theta Method

Assimakopoulos and Nikolopoulos first introduced the Theta method in 2000 [38], later than most of the classical forecasting methods still used today. Since it proved to be the the best general-purpose method on the M3 competition [134] and contained complex algebraic operations, Hyndman and Billah analysed it in detail and presented a clearer, simpler form of the method in [98], equivalent to the original.

The Theta method is unveiled in [98] to be a special case of SES, a method for smoothing data for presentation or forecasting. It models the point as the average of a historical window where each point is multiplied by an exponentially decreasing parameter. A graph of two time series – the original and its exponential smoothing is shown in Fig. 3.3. For a detailed description of SES see e.g. [55]. The SES represents a series X_t as:

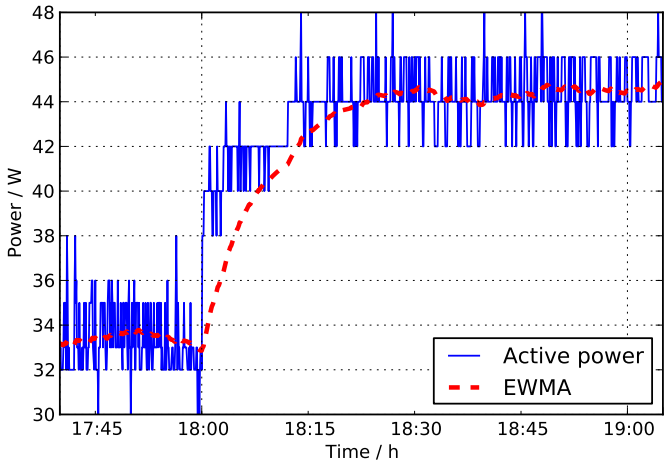


Figure 3.3: An illustration of simple exponential smoothing on a time series of server power consumption samples (obtained from the author's experimental testbed).

$$s_1 = x_0 \tag{3.3}$$

$$s_t = \alpha x_{t-1} + (1-\alpha)s_{t-1},\ t > 1 \tag{3.4}$$

where $0 < \alpha < 1$ determines how much older values influence future values and can be fitted based on the training dataset. This substitution of the original series produces an exponentially-weighted moving average, which can be seen by expressing s_t using only the original series elements:

$$s_t = \alpha \sum_{i=1}^{t-1} \left[(1-\alpha)^{i-1} x_{t-i}\right] + (1-\alpha)^{t-1} x_0. \tag{3.5}$$

From this expression it can be seen that the values closer to the currently estimated point carry a higher weight parameter, therefore having a stronger impact. The farther back in past we go, the less the values impact the estimation. To forecast unknown values using SES, for the first point $T+1$ the expression (3.4) can be used, where s_{t+1} is the predicted value for x_{t+1} For subsequent values, though, there is are no more observations of x_t, so a modified expression – *bootstrapping of forecasts* is applied where the last known observation x_T is always used:

$$s_{t+1} = \alpha x_T + (1-\alpha)s_t,\ t > T \tag{3.6}$$

Coming back to the example of daily temperature values in Kraków from Table 3.1, we can apply the formula for SES to calculate the model. According to (3.3) $s_0 = x_0 = 31$. For

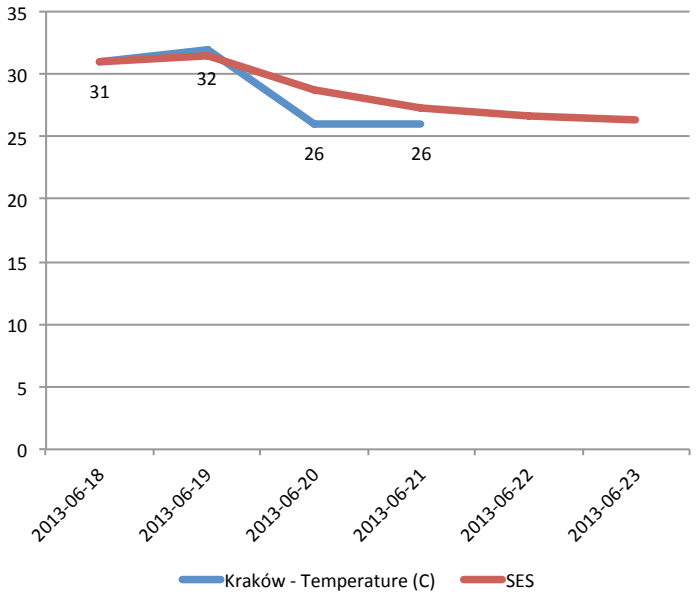


Figure 3.4: Simple exponential smoothing applied on the example from Table 3.1.

subsequent cases, expanding (3.4) we get $s_1 = 31.5$, $s_2 = 31.5$ etc. By applying bootstrapping of forecasts, we can start predicting unknown values (taking the last known observation 26 *C*): $s_5 = 26.7$, $s_6 = 26.3\ldots$ This example is shown in Fig. 3.4. Additionally, as determined by [98], the Theta method adds a random drift of the form:

$$s'_t = s_t + e_t \tag{3.7}$$

where s(t) is the deterministic SES model and e_t is a zero-long-run-mean stationary random variable.

3.1.2 Evaluation

In this section we will give a short description of the procedures applied to evaluate and compare different forecasting methods in the M3 competition [134] and in our own simulation where we used data important for managing cloud resources.

The M3 Competition

The 3003 series of the M3-Competition were selected on a quota basis to include various types of time series data (micro, industry, macro, ...) and different time intervals between successive observations (yearly, quarterly, ...). A minimum number of observations for each series was set to 14 observations for yearly series, 16 for quarterly, 48 for monthly and 60 for other series – this

Time interval	Types of time series data						
	Micro	Industry	Macro	Finance	Demographic	Other	Total
Yearly	146	102	83	58	245	11	645
Quarterly	204	83	336	76	57		756
Monthly	474	334	312	145	111	52	1428
Other	4			29		141	174
Total	828	519	731	308	413	204	3003

Table 3.2: The classification of the 3003 time series used in the M3-Competition (source [134])

measure ensured that there was enough data to develop each forecasting model. Table 3.2 lists the number of series per domain.

To compare the performance of each method to some well-known method, the Naïve2 method was chosen as the benchmark method. It is a random walk model that is applied to seasonally adjusted data by assuming that seasonality is known. The other benchmark method to compare the results against (see Fig. 3.5) was the Dampen Trend Exponential Smoothing [84].

Forecasting Cloud-related Data

To test the performance of the described forecasting methods in the context of cloud computing, we built two time series datasets. The first dataset consisted of hourly electricity prices available as a RTEP option through Ameren in Illinois, USA. They also provide historical data which we used for our evaluation [22]. The dataset consists of hourly prices for a period of 122 days, starting at 2012-05-06 00:00. The temperature data was obtained from the National Climatic Data Center [144], spanning the same time period as the electricity price data. Chicago, Illinois, USA was chosen as the location.

Measuring the Error

The methods evaluated in the M3 competition were used to make the following number of forecasts beyond the available data available to them: six for yearly, eight for quarterly, 18 for monthly and eight for other data. For cloud-related data, we did not perform out-of-sample forecasting to measure the accuracy, but instead evaluated the forecast values against the observations on the the whole dataset. Time series forecasting methods are evaluated by comparing the n forecast values (F_i) obtained from the fitted model with the actual values that continue the series (A_i), where $i \in \{1 \dots n\}$. From these values we can calculate the Mean Absolute Percentage Error (MAPE):

$$M = \frac{1}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right| \tag{3.8}$$

In an ideal case of correctly forecasting all the values, we would have a MAPE of 0. As the number and amount of errors increase, MAPE becomes larger.

Method	Average MAPE	# obs
Naïve2	15.47	3003
Theta	13.01	3003
AAM	14.63	2184

Table 3.3: Average symmetric MAPE: Naïve2 (benchmark), Theta and AAM methods (source [134])

3.1.3 Results

We will now present the results of the M3 competition and of our own simulation of forecasting cloud-related time series.

M3 Comparison

Table 3.3 lists the average MAPE values obtained after evaluating the Theta and AAM method on the M3 competition dataset [134]. The benchmark method Naïve2 is included to give some perspective to the difference in results. We can see from the results that the Theta method gives substantially better results on the large scale of several thousands time series included in the M3 dataset. Note that the AAM method is evaluated on only 2184 series – the reason for this is that yearly series do not have a history long-enough to be modelled by ARIMA, so they would corrupt the results as explained in [143].

Fig. 3.5 shows the average symmetric MAPE compared to the Dampen benchmark method for *all methods* that took part in the M3 competition. Different horizons (types of series) are grouped under labels 1, 6, 12 and 18. Taking a finer-grained look at the results, the AAM method outperformed the Theta method in horizon 6, which consists of financial monthly data. This kind of high-frequency data usually offers a larger history to learn from, which is beneficial for fitting complex models, such as ARIMA. This shows that for certain kinds of data it still makes sense to consider the AAM method.

Cloud-related Data Forecasting Precision

The MAPE values for applying AAM and the Theta method on electricity prices and temperature values are shown in Fig. 3.6. Unlike the M3 competition, the AAM method proved to be more precise on both datasets. It resulted in a significantly lower MAPE value for the electricity price time series, meaning that it was able to better fit its observations. The absolute values are somewhat better than that of the M3 competition, but that can be attributed to not using out-of-sample testing, but instead having a unified testing and training dataset. A visualisation of the forecasts of electricity price and temperature data using the more successful AAM method with its confidence intervals are shown in Fig. 3.7 and Fig. 3.8 respectively.

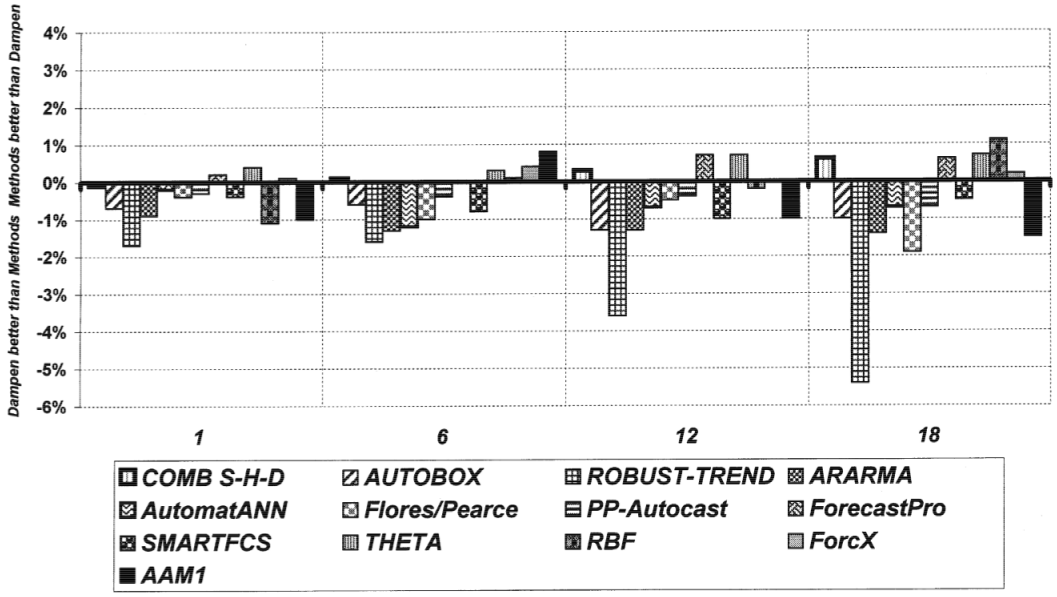


Figure 3.5: Average symmetric MAPE compared to the Dampen benchmark method (source [134])

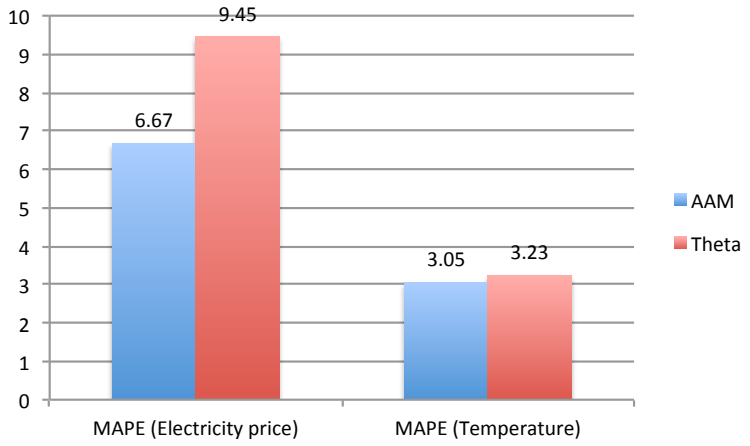


Figure 3.6: MAPE values of forecasting electricity prices and air temperature time series using AAM and the Theta method.

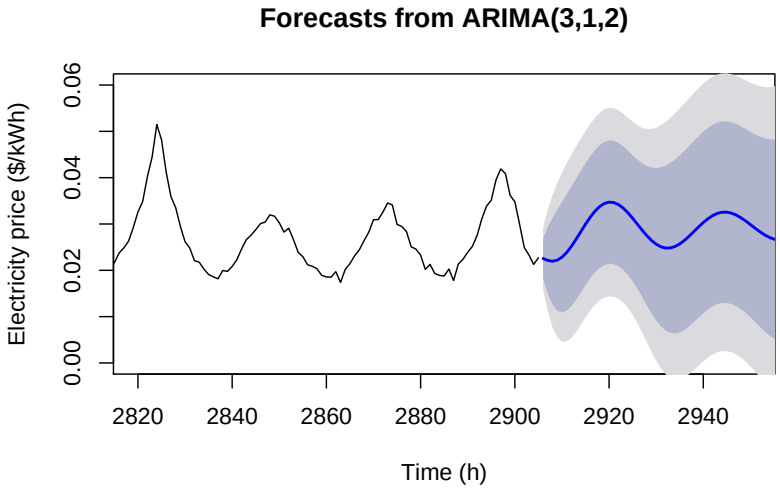


Figure 3.7: Forecasts of electricity prices with confidence intervals using AAM.

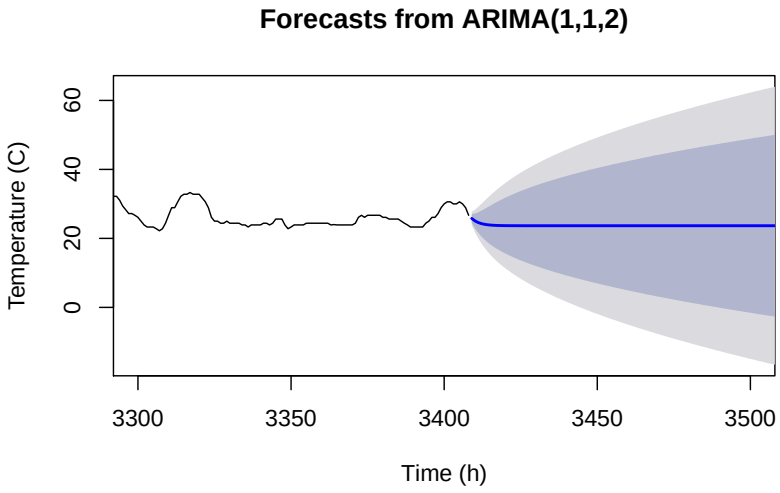


Figure 3.8: Forecasts of air temperature with confidence intervals using AAM.

3.2 Emission Trading Implications on Cloud Computing

The energy produced to power the ICT industry (and data centers constitute a major energy consumption portion) is responsible for 2% of all the carbon dioxide equivalent (CO_2e – greenhouse gases normalized to carbon dioxide by their environmental impact) emissions [1], thus accelerating global warming [10].

Cloud computing facilitates users to buy computing resources from a cloud provider and specify the exact amount of each resource (such as storage space, number of cores etc.) that they expect through a Service Level Agreement (SLA)[2]. The cloud provider then honors this agreement by providing the promised resources to avoid agreement violation penalties (and to keep the customer satisfied to continue doing business). However, cloud providers are usually faced with the challenge of satisfying promised SLAs and at the same time not wasting their resources as a user very rarely utilizes computing resources to the maximum [40].

In order to fight global warming, the Kyoto protocol was established by the United Nations Framework Convention on Climate Change (UNFCCC or FCCC). The goal is to achieve global stabilisation of greenhouse gas concentrations in the atmosphere at a level that would prevent dangerous anthropogenic interference with the climate system [11]. The protocol defines control mechanisms to reduce CO_2e emissions by basically setting a market price for such emissions. Currently, flexible models for CO_2e trading are developed at different organizational and political level as for example at the level of a country, industry branch, or a company. As a result, keeping track of and reducing CO_2e emissions is becoming more and more relevant after the ratification of the Kyoto protocol.

In this section we propose a CO_2e-trading model for transparent scheduling of resources in cloud computing adhering to the Kyoto protocol guidelines [75], we present a conceptual model for CO_2e trading compliant to the Kyoto protocol's emission trading scheme. We consider an *emission trading market (ETM)* where *credits for emission reduction (CERs)* are traded between data centers. Based on the positive or negative *CERs* of the data center, a cost is set for the environmental impact of the energy used by applications. Thereby, a successful application scheduling decision can be made after considering the (i) energy costs, (ii) CO_2e costs and (iii) SLA violation costs. Second, we propose a *wastage-penalty* model that can be used as a basis for the implementation of Kyoto protocol-compliant scheduling and pricing models. Finally, we discuss potential uses of the model as an optimisation heuristic in the resource scheduler.

3.2.1 Applying the Kyoto Protocol to Clouds

The Kyoto protocol [88] commits involved countries to stabilize their greenhouse gas (GHG) emissions by adhering to the measures developed by the United Nations Framework Convention on Climate Change (UNFCCC) [11]. These measures are commonly known as *the cap-and-trade system*. It is based on setting national emission boundaries – caps, and establishing international emission markets for trading emission surpluses and emission deficits. This is known as *certified emission reductions* or *credits for emission reduction* (CERs). Such a trading system rewards

[2]We consider the traditional business model where the desired specifications are set in advance, as is still the case in most IaaS clouds.

countries which succeeded in reaching their goal with profits from selling CERs and forces those who did not to make up for it financially by buying CERs. The European Union Emission Trading System (EU ETS) is an example implementation of an emission trading market [4]. Through such markets, CERs converge towards a relatively constant market price, same as all the other tradable goods.

Individual countries control emissions among their own large polluters (individual companies such as power generation facilities, factories...) by distributing the available caps among them. In the current implementation, though, emission caps are only set for entities which are responsible for more than 25 MtCO_2e/year [3]. This excludes individual data centers which have a carbon footprint in the ktCO_2e/year range [23].

It is highly possible, though, that the Kyoto protocol will expand to smaller entities such as cloud providers to cover a larger percentage of polluters and to increase the chance of global improvement. One such reason is that currently energy producers take most of the weight of the protocol as they cannot pass the responsibilities on to their clients (some of which are quite large, such as data centers). In 2009, three companies in the EU ETS with the largest shortage of carbon allowances were electricity producers [5]. Another indicator of the justification of this forecast is that some cloud providers, such as Google already participate in emission trading markets to achieve carbon neutrality [6].

For this reason, we hypothesize in this section that cloud providers are indeed part of an emission trading scheme and that CO_2e emissions have a market price.

3.2.2 Wastage-Penalty Balance in a Kyoto-Compliant Cloud

In this section we present our CO_2e-trading model that is to be integrated with cloud computing. We show how an economical balance can be found in it. Lastly, we give some discussion as to how such information might be integrated into a scheduler to make it more energy and cost efficient.

The CO_2e-Trading Model

The goal of our model is to integrate the Kyoto protocol's CO_2e trading mechanism with the existing cloud computing service-oriented paradigm. At the same time we want to use these two aspects of cloud computing to express an economical balance function that can help us make better decisions in the scheduling process.

The model diagram in Fig. 3.9 shows the entities in our model and their relations. A cloud offers some computing resources as services to its users and they in turn pay the cloud provider for these services. Now, a cloud is basically some software running on machines in a data center. To operate, a data center uses electrical energy that is bought from an energy producer. The energy producers are polluters as they emit CO_2e into the atmosphere. As previously explained, to mitigate this pollution, energy producers are bound by the Kyoto protocol to keep their CO_2e emissions bellow a certain threshold and buy CERs for all the excess emissions from other entities that did not reach their caps yet over the emission trading market (ETM). This is illustrated by getting negative CERs (-CERs) for CO_2e responsibilities and having to buy the same amount of positive CERs (+CERs) over the ETM. It does not make any real difference for our model if an

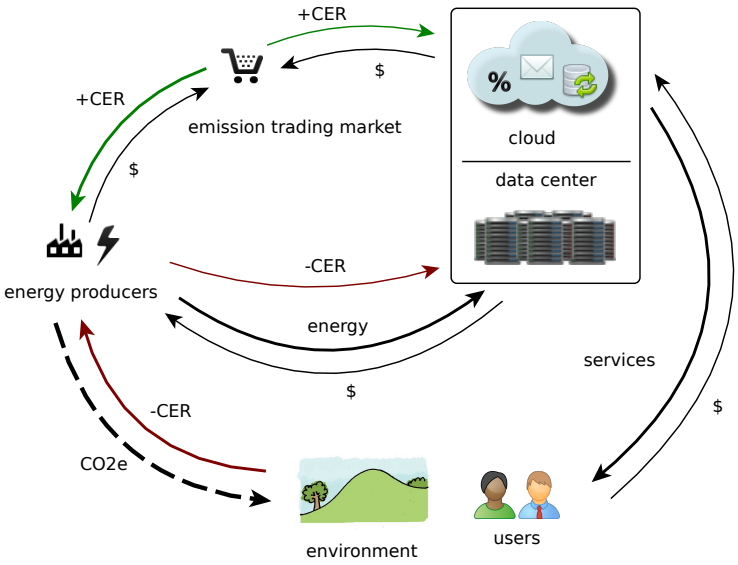


Figure 3.9: Cloud computing integrated with the Kyoto protocol's emission trading scheme

entity reaches its cap or not, as it can sell the remaining CO_2e allowance as CERs to someone else over the ETM. Most importantly, this means that *CO_2e emissions an entity is responsible for have a price*.

The other important thing to state in our model is that *CO_2e emission responsibilities for the energy that was bought is transferred from the energy producer to the cloud provider*. This is shown in Fig. 3.9 by energy producers passing some amount of -CERs to the cloud provider along with the energy that was bought. The cloud provider then has to buy the same amount of +CERs via the ETM (or he will be able to sell them if he does not surpass his cap making them equally valuable).

The consequences of introducing this model are that three prices influence the cloud provider: (1) energy cost; (2) CO_2e cost; (3) service cost. To maximize profit, the cloud provider is motivated to decrease energy and CO_2e costs and maximize earnings from selling his service. Since the former is achieved by minimizing resource usage to save energy and the latter by having enough resources to satisfy the users' needs, they are conflicting constraints. Therefore, an economical balance is needed to find exactly how much resources to provide.

The service costs are much bigger than both of the other two combined (that is the current market state at least, otherwise cloud providers would not operate), so they cannot be directly compared. There are different ways a service can be delivered, though, depending on how the cloud schedules resources. The aim of a profit-seeking cloud provider is to deliver just enough resources to the user so that his needs are fullfilled and that the energy wastage stays minimal. If a user happens to be tricked out of too much of the resources initially sold to him, a service violation occurs and the cloud provider has to pay a penalty price. This means that we are comparing the energy wastage price with the occasional violation penalty. This comparison is the core of our

wastage-penalty model and we will now explain how can a wastage-penalty economical balance be calculated.

The Wastage-Penalty Model for Resource Balancing

As was briefly sketched in the introduction, the main idea is to push cloud providers to follow their users' demands more closely, avoiding too much resource over-provisioning, thus saving energy. We do this by introducing additional cost factors that the cloud provider has to pay if he wastes too much resources – the energy and CO_2e costs shown in Fig. 3.9, encouraging him to breach the agreed service agreements and only provide what is actually needed. Of course, the cloud provider will not breach the agreement too much, as that could cause too many violation detections (by a user demanding what cannot be provided at the moment) and causing penalty costs. We will now expand our model with some formal definitions in the cloud-user interface from Fig. 3.9 to be able to explicitly express the wastage-penalty balance in it.

We assume a situation with one cloud provider and one cloud user. The cloud provides the user with a single, abstract resource that constitutes its service (it can be the amount of available data storage expressed in GB, for example). To provide a certain amount of this resource to the user in a unit of time, a proportional amount of energy is consumed and indirectly a proportional amount of CO_2e is emitted. An example resource scheduling scenario is shown in Fig. 3.10. An SLA was signed that binds the cloud provider to provide the user a constant resource amount, r_{agreed}. The cloud provider was paid for this service in advance. A user uses different resource amounts over time. At any moment the R_{demand} variable is the amount required by the user. To avoid over-provisioning the provider does not actually provision the promised resource amount all the time, but instead adapts this value dynamically, $r_{provisioned}$ is the resource amount allocated to the user in a time unit. This can be seen in Fig. 3.10 as $r_{provisioned}$ increases from t_1 to t_2 to adapt to a sudden rise in R_{demand}.

As we can not know how the user's demand changes over time, we will think of R_{demand} as a random variable. To express R_{demand} in an explicit way, some statistical method would be required and research of users' behaviour similar to that in [40] to gather real-life data regarding cloud computing resource demand. To stay on a high level of abstraction, though, we assume that it conforms to some statistical distribution and that we can calculate its mean $\overline{R}_{demand}$ and its maximum $max(R_{demand})$. To use this solution in the real world, an appropriate distribution should be input (or better yet – one of several possible distributions should be chosen at runtime that corresponds to the current user or application profile). We know the random variable's expected value E and variance V for typical statistical distributions and we can express $\overline{R}_{demand}$ as the expected value $E(R)$ and $max(R_{demand})$ as the sum of $E(R)+V(R)$ with a limited error.

Let us see how these variables can be used to model resource wastage costs. We denote the energy price to provision the whole r_{agreed} resource amount per time unit c_{en} and similarly the CO_2e price c_{co_2}. By only using the infrastructure to provision an amount that is estimated the user will require, not the whole amount, we save energy that would have otherwise been wasted and we denote this evaded wastage cost $c_{wastage}$. Since $c_{wastage}$ is a fraction of $c_{en}+c_{co_2}$, we can use a percentage w to state the percentage that is wasted:

$$c_{wastage} = w * (c_{en} + c_{co_2}) \tag{3.9}$$

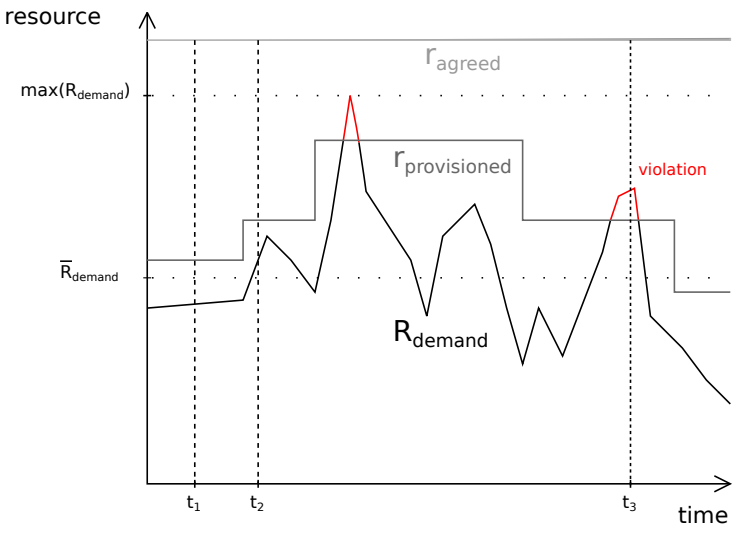


Figure 3.10: Changes in the *provisioned* and *demand* resource amounts over time

We know what the extreme cases for w should be – 0% for provisioning approximately what is needed, $\overline{R}_{demand}$; and the percentage equivalent to the ratio of the distance between $\overline{R}_{demand}$ and r_{agreed} to the total amount r_{agreed} if we provision r_{agreed}:

$$w = \begin{cases} 1 - \frac{\overline{R}_{demand}}{r_{agreed}}, & \text{if } r_{provisioned} = r_{agreed} \\ 0, & \text{if } r_{provisioned} = \overline{R}_{demand} \end{cases} \tag{3.10}$$

We model the distribution of w between these extreme values using linear interpolation: average resource utilization - a ratio of the average provisioned resource amount ($\overline{r}_{provisioned}$) and the promised resource amount ($r_{promise}$):

$$w = \frac{r_{provisioned} - \overline{R}_{demand}}{r_{agreed}} \tag{3.11}$$

If we apply 3.11 to 3.9 we get an expression for the wastage cost.:

$$c_{wastage} = \frac{r_{provisioned} - \overline{R}_{demand}}{r_{agreed}} * (c_{en} + c_{co_2}) \tag{3.12}$$

Let us now use a similar approach to model penalty costs. If a user demands more resources than the provider has provisioned, an SLA violation occurs. The user gets only the provisioned amount of resources in this case and the provider has to pay the penalty cost C_{penal}. While c_{en} and c_{co_2} can be considered constant for our needs, C_{penal} is a random variable, because it depends on the user's behaviour which we can not predict with 100% accuracy, so we will be working with $E(C_{penal})$, its expected value.

$E(C_{penal})$, the expected value of C_{penal} can be calculated as:

$$E(C_{penal}) = p_{viol} * c_{viol} \tag{3.13}$$

where c_{viol} is the constant cost of a single violation (although in reality probably not all kinds of violations would be priced the same) and p_{viol} is the probability of a violation occurring. This probability can be expressed as a function of $r_{provisioned}$, r_{agreed} and R_{demand}, the random variable representing the user's behaviour:

$$p_{viol} = f(\overline{r}_{provisioned}, r_{promise}, R_{demand}) \tag{3.14}$$

Again, same as for $c_{wastage}$, we know the extreme values we want for p_{viol}. If 0 is provisioned, we have 100% violations and if $max(R_{demand})$ is provisioned, we have 0% violations:

$$p_{viol} = \begin{cases} 100\%, & \text{if } r_{provisioned} = 0 \\ 0\%, & \text{if } r_{provisioned} = max(R_{demand}) \end{cases} \tag{3.15}$$

and if we assume a linear distribution in between we get an expression for the probability of violations occuring, which is needed for calculating the penalty costs:

$$p_{viol} = 1 - \frac{r_{provisioned}}{max(R_{demand})} \tag{3.16}$$

Now that we have identified the individual costs, we can state our goal function. If the cloud provider provisions too much resources the $c_{wastage}$ wastage cost is too high. If on the other hand he provisions too little resources, tightens the grip on the user too much, the $E(C_{penal})$ penalty cost will be too high. The economical balance occurs when the penalty and wastage costs are equal - it is profitable for the cloud provider to breach the SLA only up to the point where penalty costs exceed wastage savings. We can express this economical balance with the following equation:

$$c_{wastage} = E(C_{penal}) + [\text{customer satisfaction factor}] \tag{3.17}$$

The *[customer satisfaction factor]* could be used to model how our promised-provisioned manipulations affect the user's happiness with the quality of service and would be dependant of the service cost (because it might influence if the user would be willing to pay for it again in the future). For simplicity's sake we will say that this factor equals 0, getting:

$$c_{wastage} = E(C_{penal}) \tag{3.18}$$

Now, we can combine equations 3.12, 3.18, 3.13 and 3.16 to get a final expression for $r_{provisioned}$:

$$r_{provisioned} = \frac{max(R_{demand}) * \left[\overline{R}_{demand} * (c_{en} + c_{co_2}) + r_{agreed} * c_{viol}\right]}{max(R_{demand}) * (c_{en} + c_{co_2}) + r_{agreed} * c_{viol}} \tag{3.19}$$

This formula is basically *the economical wastage-penalty balance*. All the parameters it depends on are constant as long as the demand statistic stays the same. It shows how much on average should a cloud provider breach the promised resource amounts when provisioning

resources to users so that the statistically expected costs for SLA violation penalties do not surpass the gains from energy savings. Vice versa also holds – if a cloud provider provisions more resources than this wastage-penalty balance, he pays more for the energy wastage (energy and CO_2e price), than what he saves on SLA violations.

Heuristics for Scheduling Optimisation with Integrated Emission Management

In this section we discuss a possible application of our wastage-penalty model for the implementation of a future-generation data center. Knowing the economical wastage-penalty balance, heuristic functions can be used to optimize resource allocation to maximize the cloud provider's profit by integrating both service and violation penalty prices and energy and CO_2e costs. This is useful, because it helps in the decision-making process when there are so many contradicting costs and constraints involved.

A heuristic might state: "try not to provision more than $\pm x\%$ resources than the economical wastage-penalty balance". This heuristic could easily be integrated into existing scheduling algorithms, such as [138, 137] so that the cloud provider does not stray too far away from the statistically profitable zone without deeper knowledge about resource demand profiles. The benefits of using our wastage-penalty model are:

- a new, expanded cost model covers all of the influences from Fig. 3.9
- CO_2e-trading schema-readiness makes it easier to take part in emission trading
- a Kyoto-compliant scheduler module can be adapted for use in resource scheduling and allocation solutions
- the model is valid even without Kyoto-compliance by setting the CO_2e price c_{co_2} to 0, meaning it can be used in traditional ways by weighing only the energy wastage costs against service violation penalties.

The wastage-penalty balance in 3.19 is a function of significant costs and the demand profile's statistical properties:

$$r_{provisioned} = g(max(R_{demand}), \overline{R}_{demand}, r_{agreed}, c_{en}, c_{co_2}, c_{viol}) \tag{3.20}$$

This function enables the input of various statistical models for user or application demand profiles ($max(R_{demand})$ and $\overline{R}_{demand}$) and energy (c_{en}), CO_2e (c_{co_2}) and SLA violation market prices (c_{viol}). With different input parameters, output results such as energy savings, environmental impact and SLA violation frequency can be compared. This would allow cloud providers and governing decision-makers to simulate the effects of different scenarios and measure the influence of individual parameters, helping them choose the right strategy.

3.3 Summary

In this chapter we offered some insights into the dynamic variables affecting geographically distributed cloud systems and analysed time series forecasting as a tool for predicting their future

behaviour. We compared two time series forecasting methods – the statistically complex AAM technique consisting of an expert system that performs various data transformations to build a stationary process representing the series' temporal behaviour; and the Theta method, based on SES with random drift, relying on simpler algebraic expressions to build a geometrically-intuitive smoothed model. An empirical evaluation of both methods on a large dataset of thousands of different kinds of time series was performed to find the best forecasting methods in [134], which allowed us to see how both methods perform in the same situations. Additionally, we evaluated the methods on datasets relevant to data centers – real time electricity prices [22] and outside air temperature values [144].

The Theta method had the smallest average MAPE, therefore "winning" the M3 competition. The results of this competition have shown that simple and straightforward methods often outperform complex, theoretically advanced models on the average. This means that simple methods, such as the Theta method, represent good general-purpose tools that can be used if we have little knowledge of the domain or the art and theory of forecasting. Still, in areas where high forecasting precision is important, it would be good to collect a larger training dataset and apply more complex methods, such as AAM, to test if they would maybe perform better.

In our evaluation on cloud-related data, the AAM method proved to be better in forecasting both electricity prices and temperatures. It also proved to be better in the M3 competition when there is a long history available, as is the case with financial monthly data.

An interesting family of methods which also performed well in the M3 competition were those that combined several models to automatically choose the best one for the data at hand. These hybrid methods that take the best of both worlds are probably the future of forecasting in our more and more data-driven world.

We have shown that time series forecasting can indeed give insights about the future behaviour of dynamic variables. Such methods could find many uses in improving the management of computational resources in geographically distributed data centers forming the backbone of cloud computing.

In this chapter we also presented an approach for Kyoto protocol-compliant modelling of data centers where CO_2e also become a cost component, similar to electricity prices. We presented a conceptual model for CO_2e trading compliant with the Kyoto protocol's emission trading scheme. We consider an *emission trading market (ETM)* where CO_2e obligations are forwarded to data centers, involving them in the trade of credits for emission reduction (CERs). Such measures would ensure a CO_2e equilibrium and encourage more careful resource allocation inside data centers. To aid decision making inside this CO_2e-trading system, we proposed a *wastage-penalty* model that can be used as a basis for the implementation of Kyoto protocol-compliant scheduling and pricing models.

CHAPTER 4

Experimental VM Scheduling for Real-Time electricity Pricing

In this chapter we set the foundations of a grid-conscious cloud, taking a more informed view of the electrical grid by analysing real-time electricity prices. We propose a scheduling algorithm that predicts electricity price peaks and throttles energy consumption by pausing virtual machines. We empirically evaluate the approach on the OpenStack cloud manager in an experimental implementation and show reductions in energy consumption and costs. Finally, we define green instances in which cloud providers can offer such services to their customers under better pricing options.

We first give some insights into seasonal patterns in real-time pricing options in Section 4.1.1. In Section 4.1.2 we define green instances and state the peak pauser scheduling algorithm, with a rundown of its evaluation procedure in Section 4.2 and the actual results are presented in Section 4.3.

4.1 Foundations of the Grid-conscious Cloud

The grid-conscious cloud model is illustrated in Fig 4.1. Assuming a real-time electricity pricing model, the price of electricity offered by the utility changes hourly. Various factors such as active generation pools, electricity markets, volatile demand [177, 178], and CO_2e emission legislation and markets from Chapter 3 that influence the electrical grid are condensed in this price signal. Combined with the elasticity of cloud computing, this creates an opportunity to optimize the conversion of energy to computation that would benefit both sides.

The grid-conscious cloud is based on the idea that *computation is more elastic than energy.* Moving energy around, storing it or building infrastructure to satisfy demand can be very expensive and environmentally inefficient [177]. Computation, on the other hand – especially using modern paradigms such as virtualisation and cloud computing, can be scaled outwards or

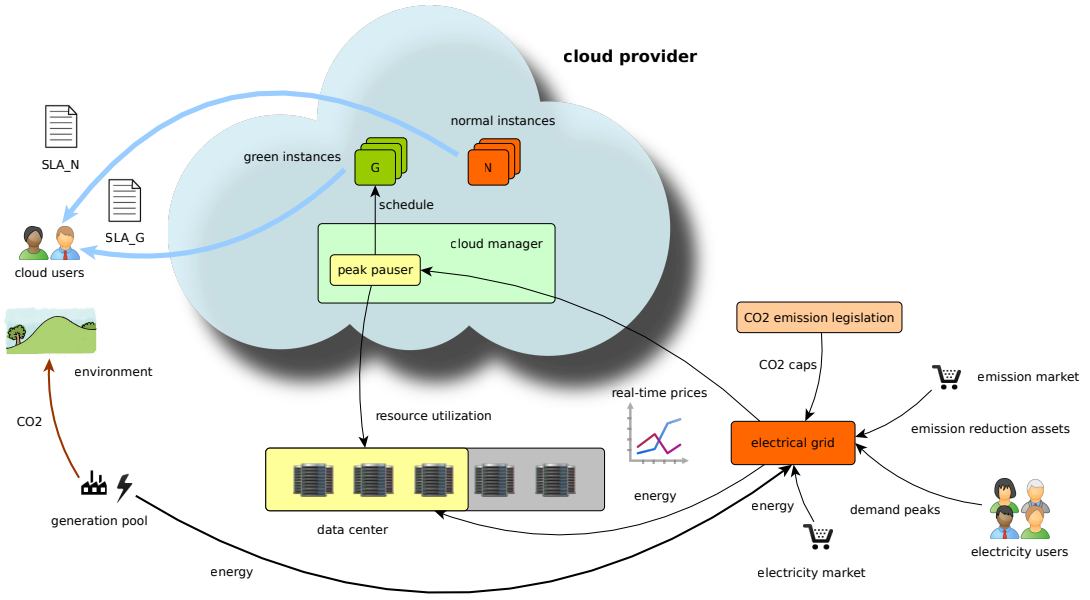


Figure 4.1: The grid-conscious cloud model.

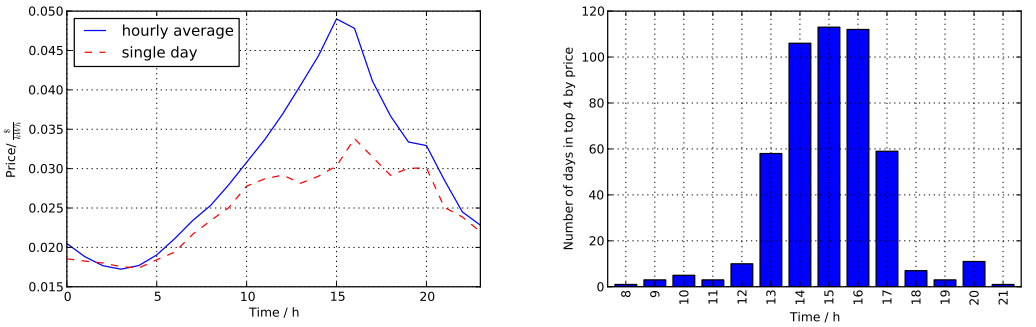
inwards, moved to other locations and postponed for a later time, depending on the needs of users and resource providers.

The provider of a grid-conscious cloud is motivated to weigh the electricity price parameter in its scheduling and resource utilization process. It needs a scheduler that can respond to changing electricity prices, altering the cloud's physical resource usage and energy demand. Changing resource usage potentially affects end users so it might be necessary to open a new business interface towards the end users. Taking all this into account, the grid-conscious cloud needs to satisfy three major requirements which we examine in the following subsections: (1) a real-time pricing option of buying electricity, (2) a resource scheduler that considers electricity prices to throttle energy demand when they are high and (3) a business model to offer this kind of service to end users with a justification for potential performance degradation.

4.1.1 Patterns in Real-Time Electricity Prices

Peak-load pricing is a technique where demands for a public good in different periods of the day, month or year are considered to find the optimal capacity and the optimal peak-load prices. The method was being developed as far back as 1949 [51]. This is the basis of electricity spot markets, where electricity can be bought at real-time prices [163].

As discussed in Chapter 3, many utilities (such as Ameren [22]) offer a real-time pricing (RTP) option where electricity prices change hourly, based on market supply and demand. Generally



(a) Peak hour distribution

(b) Peak electricity price hours distribution

Figure 4.2: Historical electricity prices (data taken from [22])

speaking, market prices are highest during times of peak demand (during daytime, usually peaking at 15:00), as shown in Fig. 4.2a. In RTP, customers are given price signals to guide their energy use and ease the burden on utilities during peak hours. Such customers are rewarded by potentially saving money when compared to the standard rate. There exist other pricing options, such as *critical peak pricing* and *time of use pricing* which could be suitable for our grid-conscious cloud model. In this study, however, we focus only on RTP – considered to be the most direct and efficient demand-response program [32].

An important point to consider that was analysed in [108] is that in some cases utility companies have to resort to environmentally more harmful energy generation methods to handle peak demand. This is true in the UK, for example, where the National Grid can call upon about 2 GW of diesel-generated power to meet demand spikes, resulting in hundreds of hours of expensive and harmful fuel usage annually [36]. The result is that a high electricity price also indicates a high rate of CO_2e emissions. This proportionality is further fortified under the Kyoto protocol's cap-and-trade model, where the CO_2e emissions themselves carry a market price as we analysed in Chapter 3. This additional CO_2e emission price increases the price of energy coming from environmentally harmful sources even more. These specifics vary between different locations, however, which underlines the importance of having access to information grid through smart grids to make more informed, efficient decisions.

Another viewpoint of real-time prices is that it encourages polite behaviour, complying with the utilities' needs. This enables utilities to be more efficient (in terms of cost efficiency, reliability and environmental impact) and is said to create *energy reduction assets* [13]. Such assets can then be exchanged for certain other benefits, such as lower energy prices, meaning that they benefit both sides. These power-reduction assets can be achieved more directly by integrating our grid-conscious cloud model with a direct demand/response control mechanism available in modern smart grids [90].

These examples show that there exist both monetary and ecological incentives for reducing energy consumption during hours of electricity price peaks – especially in computing clouds, which are substantial energy consumers [111].

4.1.2 Optimizing the Scheduler: the Peak Pauser Algorithm

The aim of this algorithm is to curtail energy consumption of a cloud during the hours of the day with the highest electricity price. Its pseudo-code is shown in Alg. 1.

The algorithm first finds n hours in a day that are statistically most probable to be the peak-price hours. This number is defined using the parameter $downtime\_ratio$ given to the algorithm and is defined as

$$downtime\_ratio = \frac{\mathrm{n}}{24} \tag{4.1}$$

We then define the set of *expensive hours* by calculating hourly price averages over the provided historical dataset, sorting them descending and selecting the first n hours in the $find\_expensive\_hours$ function that requires historical electricity price data such as [22].

We verified this function by analysing how often certain hours of the day appear among the top 4 by price (Fig. 4.2b shows the regular cyclic nature with peaks in the afternoon) and found it to result in only a negligible error [1] compared to an ideal scenario where we know the prices in advance, sufficiently good for the purposes of our discussion.

Knowing how to determine expensive hours, we can define the scheduling algorithm as an endless loop (the *peak_pauser* function) that checks if the electricity is predicted to be expensive at the moment (in the *is_expensive* function that returns a boolean value depending on the current time's membership in the *expensive_hours* set). If the price is expensive, the scheduler pauses the set G of instances it controls (green instances, which we will further discuss in the next section) and if it is not expensive, instances in G are unpaused (or left running if they were not paused before). The scheduler can then remain idle for the remainder of the hour, so as not to waste resources.

A possible alternative to pausing would be to switch to battery power supply [148, 49] during expensive hours. Additional logic can be added to the algorithm by dynamically determining duration of the pause interval (the parameter $downtime\_ratio$), based on e.g. considering the current day's deviation from monthly or annual averages. This way we could have longer pause periods during unusually "expensive" days and close-to-normal operation on "cheaper" days. For the purposes of our evaluation, we chose a predefined value $downtime\_ratio = 0.16$, yielding in 4 paused hours.

4.1.3 The Green Instance Model

To justify VM pausing to users, we propose *green instances* as an option in the manner of Amazon's *spot instances* [29]. Spot instances have a disadvantage – they will only be running when there are free computing resources; and an advantage – a more affordable price. Expanding on this philosophy of acceptable trade-offs, we devised the green instance model. We envision it as an option where (1) compute instances are offered at a reduced availability time, but (2) at a lower price and (3) with environmental metrics presented to the user. From the cloud provider's perspective, this allows for more flexibility while scheduling to reduce energy costs and lessen the environmental impact which is good for public relations.

[1] The root-mean-square error of the sum of expensive hours in a day determined by our function compared to an optimal daily-changing set that assumes a priori price knowledge equals 0.0058 $\$/kWh$ or $\approx 3\%$ of the absolute amounts.

Algorithm 1 The peak pauser algorithm.

```
function FIND_EXPENSIVE_HOURS(downtime_ratio)
    prices ← historical hourly prices
    avg_prices ← group prices by hour and calculate mean
    sort avg_prices descending by price
    n ← ceil(downtime_ratio * 24)          ▷ ceil: find first larger integer
    expensive_hours ← first n elements of avg_prices
    return expensive_hours
end function

expensive_hours ← find_expensive_hours(
    downtime_ratio)
function IS_EXPENSIVE
    time ← current time of day
    return time.hour ∈ expensive_hours
end function

function PEAK_PAUSER(G)
    while True do
        if is_expensive() then
            pause ∀instance ∈ G
        else
            unpause ∀paused instance ∈ G
        end if
        idle for the remainder of the hour
    end while
end function
```

This would be an opt-in model, where users would be offered an additional SLA – SLA_G to rent green instances or choose normal instances (SLA_N). The set G of instances managed by the peak pauser would be restricted to green instances only.

Only users who applied for green instances and therefore accept occasional downtimes (at relatively predictable times of day) would feel the consequences of the peak pauser and create energy reduction assets (ERA)s. ERAs basically mean that the cloud provider gets better electricity prices in return for helping the electrical grid's efficient operation. As a result, the cloud provider can offer lower prices to the end user, to compensate for the reduced VM availability. So, where in spot instances users sacrifice performance for profitability, in green instances they sacrifice availability for profitability.

As a further reinforcing factor in favour of green instances, users could be presented with the approximated energy savings arising from choosing green over normal instances. Curry et al. [68] denote this type of information *environmental charge-backs* (*EC*). Using carbon emission

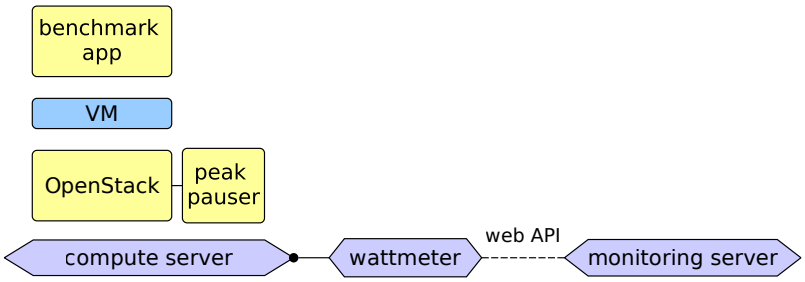


Figure 4.3: Experiment deployment

factor (CEF) and power usage efficiency (PUE) defined in [41] we could express it as:

$$EC = CEF * PUE * (\text{VM energy consumption}) \tag{4.2}$$

VMeter developed in [50] might be used to determine the energy consumption of a user's VMs or the approach from [35] for parallel applications.

Green instances would target the same user group as Amazon's spot instances do and the sole fact that Amazon, a commercial corporation, still offers this service shows that there are interested parties. As a potential use-case, many automated background processes would be fit for running on green instances. Examples of such processes are nightly builds of software (frequent and long-lasting due to compile-time optimization for various architectures), automated testing, web crawling, offline data mining etc. However, for applications that require real-time human interaction, access to data in a storage-as-a-service manner [125], normal instances would be preferred.

4.2 Evaluation Methodology

The goal of the empirical testing environment is to show how the peak pauser scheduler influences energy efficiency and application execution time. To measure the effects of applying the peak pauser scheduler in practice, it was implemented on top of the open source cloud manager OpenStack [105], hosting a VM. The server running the experiment was connected to a wattmeter that provides us with precise information about power consumption. Additionally, to evaluate our scheduler under more realistic conditions than our prototype environment, we simulated the scheduler's effects in production environments, such as that of Google [77].

We will now go through the details of our evaluation setup and show how we measured the effectiveness of the peak pauser algorithm empirically and by simulating production systems.

4.2.1 Empirical Testing

We ran a predictable synthetic benchmark application inside a VM controlled by the peak pauser for approximately 24 hours. Exactly the same experiment was performed once again, but *without a scheduler* to obtain comparison results. Two assumptions were made due to electrical price

data availability – that the data center is located in Illinois, USA which offers a real-time pricing option [22] and that the experiment occurred a couple of weeks in the past.

The parameters of the peak pauser were set to pausing for a total of 4 hours in a day ($downtime\_ratio = 0.16$). The statistically most probable peak hours were determined according to 3 months of historical electricity prices [22] before (non-inclusive) the day the experiment was assumed to be running on.

As a benchmark application we used Berserk [16], our framework for running CPU-intensive tasks – repeated recursive calculation of Fibonacci numbers. The benchmark application was run inside a VM deployed on an OpenStack [105] compute server [2] whose power consumption was measured as illustrated in Fig. 4.3. The peak pauser scheduler was run on the same physical server as OpenStack and the Philharmonic [72] framework we developed used its API to control the managed VM.

We measured *active power* consumed by the compute server using a wattmeter [3] offering a web interface for collecting data.

4.2.2 Estimating Savings in Production Systems

To broaden our evaluation by estimating savings in today's production systems, we relied on a study by Google [77] to gain insight into their power consumption during peak and low demand. According to their study, peak power consumption of a server ranges from 100 to 250 W and idle power (the power consumed when nothing is executed on this server) can be as low as 50-65% of this amount. This ratio of idle and peak power is referred to as the *idle ratio*. It represents the energy elasticity of a server and turned out to be very important in our study.

In addition to this, existing mechanisms such as suspend and wake-on-lan are already available for completely turning off under-utilized components. In fact the topic of energy-proportional servers is currently being studied extensively [140] and it is likely that energy elasticity will improve in the future, aiming for an ideal idle ratio of 0.

A synthetic power time series was generated according to a simple model of our empirically collected data and scaled to match production-quality parameters from [77]. We assumed normally distributed oscillation around the peak (during VM execution) and idle (during a pause event) power values with a variance matching our experiment. The empirical and synthetic power signals can be seen in Fig. 4.4. Power is centered around 100 W peak power during VM execution and 60 W idle power while the VM is paused. The synthetic signal was used for estimating savings under different energy elasticity parameters.

4.2.3 Calculating Electricity Costs

We previously established electricity costs to be more proportional to the actual environmental impact than the bare energy consumption, which we measure. To get as close as we can to knowing the impact our system has on the environment, we therefore want to calculate the total monetary expense. As we only have access to real low-level metrics obtained using the wattmeter

[2] AMD Opteron 4130 2.6 GHz CPU, 8 GB RAM, Ubuntu 12.04 server 64-bit GNU/Linux OS with OpenStack Essex

[3] EATON ePDU PW104MA0UC34

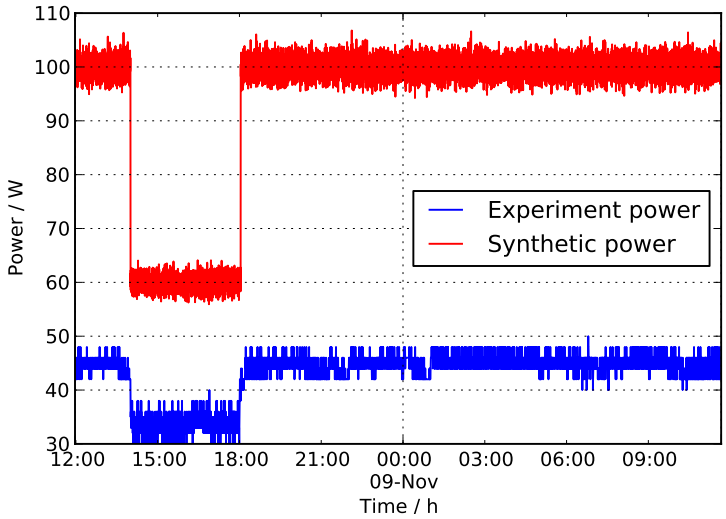


Figure 4.4: Power consumption in our experiment and a derived synthetic signal.

to monitor server behaviour in an empirical experiment, we have to calculate these monetary expenses on our own. To achieve this, we tried to mimic the pricing system actually used by the utilities in a real-time electricity price business model and we will explain these methods in detail here.

We sampled active power measurements of the compute server every 5 seconds. From this, we can calculate energy consumption and, according to real-time electricity prices [22] corresponding to the appropriate time intervals, derive a total energy price. S_{total}, the total electricity price in a time interval $T = \overline{t_0 t_N}$ is calculated from a numerical integral (we used the basic rectangle rule):

$$S_{total} = \sum_{t=t_0}^{t_N-1} \frac{t_N - t_0}{N} * P_t * C_t \tag{4.3}$$

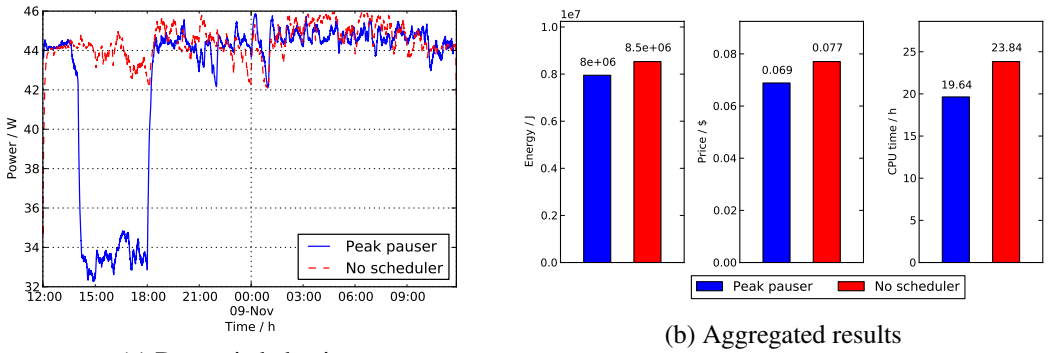
where P_t and C_t stand for power and electricity price in moment t, respectively and N is the number of samples.

4.3 Results

We will now present the results obtained after running the empirical experiment, followed by our estimation of savings in a production environment based on a synthetic power signal. Finally, we assemble the results into a green instance SLA that could be offered to users.

4.3.1 Empirical Savings

We are interested in a comparison of running the experiment for 24 hours with and without the peak pauser scheduler with regards to:



(a) Dynamic behaviour

(b) Aggregated results

Figure 4.5: Empirical evaluation results

- total consumed energy (also considering the energy consumed while the VM is paused)
- total electricity price based on dynamic, hourly charging
- benchmark CPU time

Runtime results can be seen in Fig. 4.5a. The curves are smoothed using the exponentially weighted moving average (EWMA) method [161]. They show how the compute server's power consumption changes throughout the whole experiment. The drop in power from cca. 44 to 34 W ($\approx 23\%$ or an idle ratio of $\approx 77\%$) can clearly be seen from 14 to 18 h. The VM is paused during this time interval.

The results of aggregating measurements over the entire 24 hours of the experiment are shown in Fig. 4.5b. The total energy consumption is $\approx 5.3\%$ lower than the amount consumed without a scheduler, due to the the reduced power consumption during VM pausing. The difference in the electricity price is even larger, since the peak pauser only excludes the most expensive (financially and environmentally) hours of the day, the amount spent during a single day is $\approx 6.9\%$ lower when using the peak pauser.

The CPU time the benchmark application received in the experiment is 4 hours less when using the peak pauser scheduler. This amounts to $\approx 17.6\%$ fewer actual calculations. This is a considerable performance deterioration, however it would only affect green instances whose owners agreed on fewer computing resources to increase energy efficiency and get a better price. An important thing to note here is that the current benchmark implementation only works in a single thread, not being able to consume all the CPU cores. In reality, where many processes are running, consuming many CPU cores, the ratio of energy and price savings to CPU time wastage would be greater, which we examine next.

4.3.2 Projected Savings

After applying the same energy and price calculation methods on synthetic data described in Section 4.2.2, much better savings can be achieved. Fig. 4.6 shows the aggregated results on

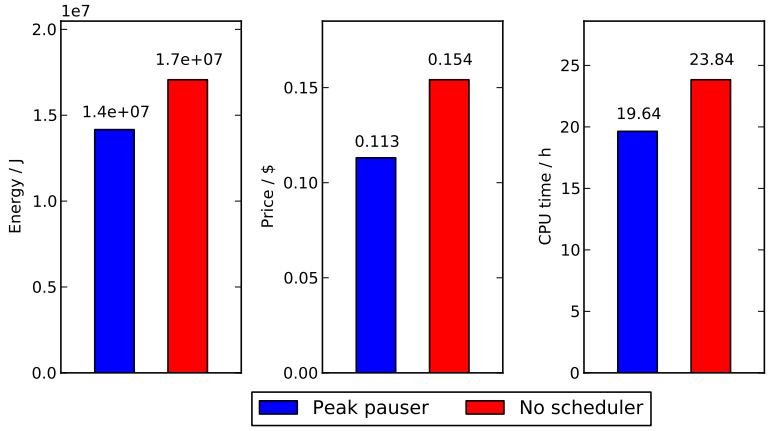


Figure 4.6: Synthetic data results

	Energy savings		Price savings	
idle ratio | P_{peak}	100 W	200 W	100 W	200 W
0%	16.96%	17.01%	26.56%	26.63%
30%	11.93%	11.94%	18.68%	18.69%
60%	6.82%	6.82%	10.67%	10.67%

Table 4.1: Estimated energy and price savings

a server with a power of 200 W during peak load and 0 W while being idle. This scenario assumes an ideally energy proportional server or the utilization of a suspend and wake-on-LAN mechanism on the physical machine (albeit a bit simplified as no delay is added to account for these actions). The raw energy savings are $\approx 17.1\%$, roughly equal to the drop in availability. The price savings exceed both of these values and amount to $\approx 26.63\%$ which is a considerable improvement.

Savings based on more combinations of peak power and idle ratio (the ratio of idle and peak power) parameters are given in Table 4.1. It is interesting to note how peak power does not seem to influence savings much – the difference between 100 and 200 W accounts for less than 1%. The idle ratio, on the other hand, has a high impact, underlining the importance of dynamically adjustable, power-proportional servers.

Based on the estimated annual electricity costs for a company as large as Google conservatively placed at $38M [155], even with a realistic power usage efficiency of 1.3 the above savings amass to very large numbers and show considerable impact – both economically and environmentally.

4.3.3 Resulting SLA

Given a 4-hour daily pause, instance availability is 83.3%. If we consider the 0–200 W scenario from Fig. 4.6, a PUE of 1.3 and a CEF of 1537.82 lb/MWh measured in [2] for Illinois, we can

calculate the annual environmental charge-back for green instances to be 1300 kgCO_2e. This is 300 kg less than a normal instance would produce (equivalent to driving an average car for 811 km). This is an approximation, but it gives an idea of the order of magnitude and as such might be presented to users as an additional advantage of green instances.

If we assume a normal instance cost of $0.060 per hour, the 26.6% savings in electricity costs would mean that green instances could be offered at $0.044 (disregarding many other factors such as equipment amortisation and maintenance costs which are out our our work's scope).

4.4 Summary

In this chapter we have shown a practical way of utilizing information about the electrical grid in the domain of cloud computing to visibly reduce energy consumption and costs.

We presented the *peak pauser* scheduling algorithm that offers a clean way of managing virtual machines in a computing cloud. It pauses computation during hours when the electricity price is statistically most probable to peak. This mechanism reduces energy costs through controllable availability reduction. Offering this kind of service as *green instances* under special SLAs to willing users only, would ensure that no harm is done from the user point of view. Furthermore, this represents environmentally friendly behaviour, because energy production applies most stress to the environment during times of peak demand, when it has to resort to faster and more inefficient generation methods.

Our prototype implementation and experimental evaluation show that savings are indeed possible in real-life systems. Results stemming from our synthetically-scaled projections of different parameters give a sketch of potentially even higher gains if the methods were to be used in production systems of today's leading cloud providers.

CHAPTER 5

Pervasive Cloud Control for Geotemporal Inputs

In this chapter, we expand the grid-conscious cloud model from the previous chapter for geographically distributed data centers and additional geotemporal inputs like temperatures affecting cooling efficiency. Distributed data center infrastructure changes the rules of cloud control, as energy costs depend on current regional electricity prices and temperatures. We denote geotemporal inputs, cloud requirements, regulations and other factors that guide the cloud provider's actions as *decision support components*. We define *forward compatiblility* as being able to cope with additional decision support components without drastic changes of the core architecture. Hence, to account for emerging technologies surrounding the cloud ecosystem, a maintainable control solution needs to be forward-compatible with new decision support components. Existing cloud controllers are focused on VM consolidation methods suitable only for a single data center or consider migration just in case of workload peaks, not accounting for all the aspects of geographically distributed data centers. In this chapter, we propose a pervasive cloud controller for dynamic resource reallocation adapting to volatile time-dependent and location-dependent factors, while considering the QoS impact of too frequent migrations and the data quality limits of time series forecasting methods, such as the methods analysed in Chapter 3. The controller is designed with extensible decision support components. We evaluate it in a simulation using historical traces of electricity prices and temperatures. By optimising for these additional factors, we estimate 28.6% energy cost savings compared to baseline dynamic VM consolidation. We provide a range of guidelines for cloud providers, showing the environment conditions necessary to achieve significant cost savings and we validate the controller's extensibility.

In the remainder of this chapter, Section 5.1 explains the research problem intuitively on a real example of geotemporal inputs and provides a high-level description of our pervasive cloud controller. The formal specification of the plug-and-play decision support components and the optimisation problem specification is presented in Section 5.2. The proof-of-concept implementation of the forward-compatible optimisation engine of the pervasive cloud controller we

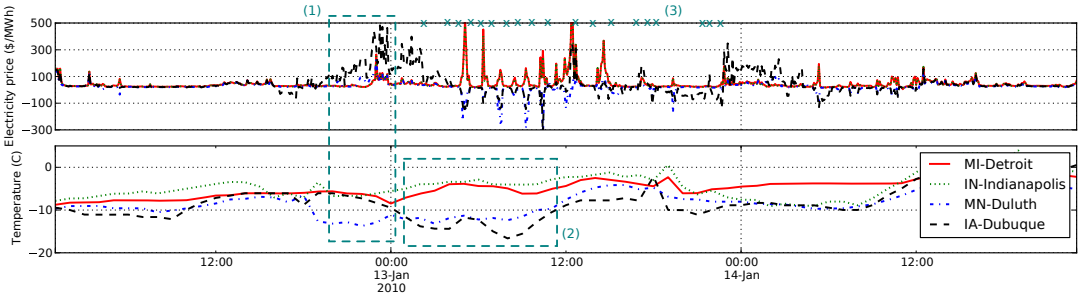


Figure 5.1: Geotemporal inputs (real-time electricity prices and temperatures) at four locations in the USA during a period of three days.

developed is explained in Section 5.3 and in Section 5.4 we describe the evaluation methodology and discuss the results.

5.1 Geotemporal Cloud Environments

We now explain the geotemporal environment surrounding geographically distributed clouds on a real example of electricity prices and temperatures. On this use case, we will give an overview of the problem on an intuitive level and give a high-level description of our pervasive cloud controller, before detailing the formal specification of the model in the following section.

5.1.1 Problem Overview

Example geotemporal inputs for four US cities are shown in Fig. 5.1. Temperature values within a single day change up to 15 degrees between peaks and lows and even larger relative differences can be observed in the volatile electricity prices. Electricity price data was obtained from [33] and temperatures from [19]. Rapid changes in geotemporal inputs can occur dynamically. The peak that can be seen in Dubuque on January 12th from 21:00 to 23:00 (1), results in five or more times the average prices. It can be observed that temperature peaks occur towards the end of the day, while lows occur during nights. Even though electricity price is more volatile, partial dependence on previous data points can be seen. This means that it is possible to model their behaviour to forecast probable future values, and in fact is done in practice [178, 19].

To explain the potential and challenges of geotemporal inputs in the context of cloud computing, let us assume that there are two data centers – one in Detroit and another one in Dubuque. For the data shown in Fig. 5.1 during the period (1), it makes sense to run more VMs in Detroit when temperatures are the same, because electricity is more expensive in Dubuque (constantly over 100 $/MWh, reaching 500 $/MWh) than in Detroit (less than 100 $/MWh). However, when it gets 10 C colder in Dubuque three hours later (2) and electricity prices become lower than in Detroit, less energy would be consumed on cooling there, resulting in lower energy costs, so it is better to migrate a number of VMs from Detroit to Dubuque and shift computational load this way.

A challenge in adapting cloud control for geotemporal inputs is that the cloud provider cannot migrate VMs between different locations too rapidly, as this wastes bandwidth, incurs an energy overhead and impacts QoS. This is underlined even more by the volatile variable behaviour observable in electricity prices. In Fig. 5.1, we marked by crosses (3) all the moments throughout January 13th when ratios between electricity prices in Detroit and Dubuque change significantly, offering an opportunity to save on energy costs by reallocating VMs using live migrations. We can see that 19 migrations would be performed this way. If we assume a downtime caused by a live VM migration to last for one minute, which is possible based on the model presented in [121], this would result in a VM availability of 98.68%. This availability is considerably lower than the 99.95% availability rates advertised by Amazon and Google in their SLA and incurs extra data transfer costs. The challenges arising from this are: (1) To profit from geotemporal inputs in cloud computing, the trade-offs of the energy savings of geotemporal inputs, the migration overheads and impact on QoS, as well as the data accuracy provided by the forecasting methods for future geotemporal inputs all have to be considered and reconciled in a long-term plan. (2) In reality, the problem has to be solved on a much larger scale with more data centers and thousands of VMs. New ways of controlling VMs across geographically distributed data centers have to be developed to address these challenges.

5.1.2 Pervasive Cloud Controller

We now present our pervasive cloud controller approach by explaining the identified requirements, defining the architecture of the solution and giving a high-level overview of its workflow.

Requirements

In our approach, we consider an IaaS cloud provider hosted on multiple geographically distributed data centers. The cloud is assumed to be operating in an environment comprising geotemporal inputs such as RTEP and temperature-dependent cooling efficiency (other inputs can be added as well). The cloud is governed by a controller system that manages virtual and physical machines in all data centers and can issue actions, such as migrating a VM from one PM to another, suspending or resuming a PM.

#CPUs	RAM	storage	availability	price
4	15 GB	80 GB	99.95%	$0.28/hour

Table 5.1: Example SLA.

The first input are user goals represented by SLAs, specifying the number of requested VMs, their resource and QoS requirements. An example of user requirements for an Amazon *m3.xlarge* VM specified in an SLA is shown in Table 5.1. The second input are the geotemporal inputs, providing time series metrics describing each of the data center locations, such as electricity price and temperature data, example values of which are shown in Fig. 5.1. Each geotemporal input is a time series of past and current values and, using time series forecasting, it is possible to

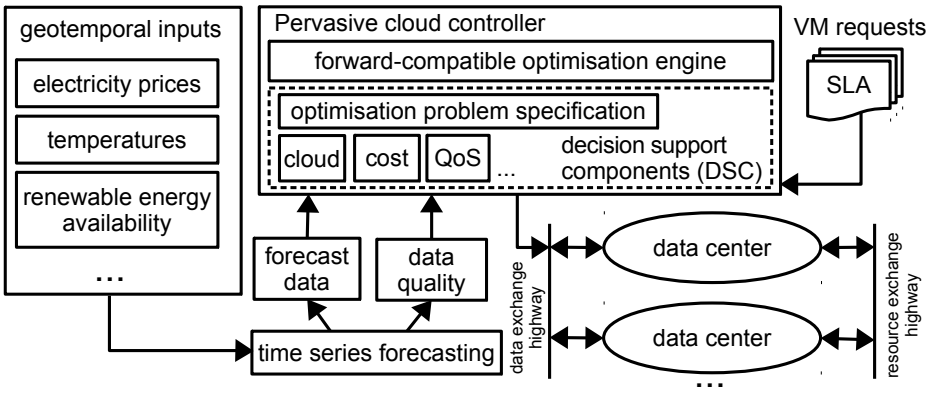


Figure 5.2: Pervasive cloud controller architecture.

predict future values and the accompanying data quality (reach and the most likely error rate). The controller's task is to output a schedule that determines where each VM is deployed and for each PM if it is running or suspended at any point in time.

Architecture

Fig. 5.2 shows the architecture of our proposed pervasive cloud controller for managing a geographically distributed cloud based on geotemporal inputs. On a high level, geotemporal inputs are used to obtain forecast data and the corresponding quality. This input is, together with the SLAs, provided to the pervasive cloud controller, which generates a long-term schedule of control actions to apply to the geographically distributed data centers.

Looking at the architecture details, geotemporal input forecasts are converted by the controller into values meaningful to the cloud provider, e.g. data center energy costs that combine RTEP with the cooling overhead, environmental impact etc. These measures along with other internal measures like cloud capacity and the QoS stemming from actions planned for VMs are all combined into an optimisation problem specification as decision support components. To support new geotemporal inputs, SLA metrics or cloud regulations, it is important for the decision support components to be extensible in a plug-and-play manner, i.e. without requiring architectural changes. We formally present the decision support component model in the following section. The role of ensuring decision support component extensibility lies in a forward-compatible optimisation engine. It considers the decision support components as criteria to plan and optimise a schedule of control actions for a future period. The challenging part of ensuring forward compatibility with new decision support components is that there has to be a separation of the schedule evaluation logic and the optimisation logic. The schedule is evaluated using geotemporal inputs and the time-based allocation of VMs to PMs, to estimate the actions' outcome in terms of costs, QoS and any other decision support components. This evaluation is then used by the controller in a black-box manner to explore the search space of possible actions using its custom optimisation logic and the high-level information about each schedule returned by the evaluation logic. The selected schedule is applied over time to physical and virtual machines in the cloud

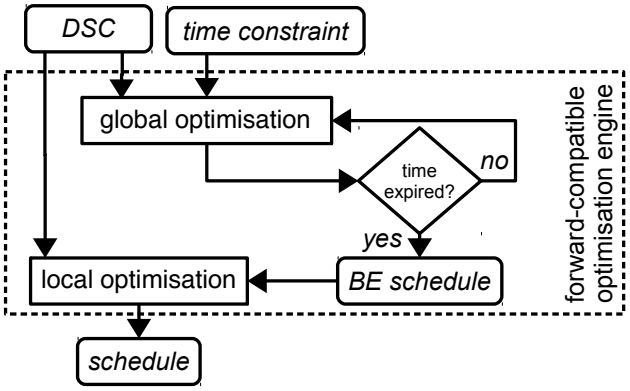


Figure 5.3: Optimisation engine workflow.

(forwarding control actions through a data exchange highway) by utilising live VM migrations as a resource exchange highway to redistribute computational load between the data centers.

Workflow

The forward-compatible optimisation engine worfklow is illustrated in Fig. 5.3. It collects the decision support components, related together as an optimisation problem specification and produces a schedule of control actions to apply in the cloud. Given that bin packing of allocating VMs to PMs is an NP-hard problem and in our case we add to it the dimension of time and time-related QoS requirements (e.g. the VM migration frequency), an optimal solution can not be found at runtime for an arbitrary problem size. To overcome this, we propose a two-stage optimisation process. The first stage is a global optimisation method that sweeps the whole search space looking for a global optimum within a time constraint (provided as an additional parameter that the cloud provider specifies). We then take the best-effort schedule this method was able to find (BE schedule) and pass it to a second stage local optimisation method that continues to improve it with a primary goal of satisfying all the hard constraints among the decision support components that the global optimisation failed to satisfy. In Section 5.3 we present our proof-of-concept implementation for the global and local optimisation methods based on a hybrid genetic algorithm with greedy local constraint satisfaction. We later use these methods to evaluate the pervasive cloud controller.

5.2 Decision Support Components

In this section we formally specify the cloud, QoS and cost decision support components and show how they are related together into an optimisation problem specification of optimisation goals and constraints. These decision support components are then used by the optimisation engine in the schedule generation.

5.2.1 Cloud and QoS Components

We consider a single IaaS cloud provider that is represented by *DCs*, a set of d geographically distributed data centers and *PMs*, a set of p physical machines it operates. We define each physical machine's location at one of the data centers.

$$\forall pm \in PMs,\ loc(pm) \in DCs \tag{5.1}$$

As we are modelling dynamic system behaviour, we define a time period in the range from t_0 to t_N (denoted $[t_0, t_N]$), of N discrete (arbitrarily small) periods.

User requirements are defined by $vmreqs_t$, a set of virtual machine requests at a moment t, each of which can either ask for a new VM to be booted or an existing one to be deleted. These events are controlled by end users and we assume no prior knowledge of the users' requests (as is the case in IaaS clouds like Amazon EC2). Based on the past and current $vmreqs_t$, we define VMs_t, a set of VMs provided to the users at time t.

The cloud provider defines an extensible set of r resource types that have to be specified through an SLA, e.g. number of CPUs and amount of RAM. The exact resources for a single VM are an ordered r-tuple of values, defining the VM's *spec*:

$$spec_{vm} = (res_{vm,i}, \forall i \in \{1, \dots r\}),\ \forall vm \in VMs_t \tag{5.2}$$

where $res_{vm,i}$ is the i-th resource's value. For the example shown in Table 5.1, there are three quantitative resource types (number of CPUs, amount of RAM and amount of storage) that are provided on the infrastructure level, so $r = 3$. Given the concrete values for an *m3.xlarge vm* instance, we have $spec_{vm} = (4, 15, 80)$. Similarly, the capacity of a PM is defined as an r-tuple of the resource amounts it has:

$$spec_{pm} = (cap_{pm,i}, \forall i \in \{1, \dots r\}), \forall pm \in PMs \tag{5.3}$$

We define the VM allocation at moment t as:

$$\forall pm \in PMs,\ alloc_t(pm) \subseteq VMs_t,\ \text{s.t. } \forall vm \in alloc_t(pm) \text{ is hosted on } pm \text{ at moment } t \tag{5.4}$$

Effectively, $alloc_t,\ \forall pm \in PMs$ is the cloud state at moment t. For any two subsequent moments t_i and $t_{i+1},\ t_i \in [t_0, t_{N-1}]$, a *vm* is considered *migrated* if $\exists pm_j, pm_k \in PMs$ s.t. $vm \in alloc_{t_i}(pm_j)$ and $vm \in alloc_{t_{i+1}}(pm_k)$. The number of such migrations for a *vm* in some relevant period specified by the cloud provider (e.g. an hour) is denoted $R_{mig}(vm)$ and represents the rate of migrations.

5.2.2 Cost Components

Progress has been made in modelling various aspects of cloud energy costs and we shortly outline the relevant findings of the existing energy-aware cost model using our notation. We then proceed with presenting our own pervasive cost model for expressing energy costs of IaaS clouds based on geotemporal inputs.

Energy-Aware Cost Model

Power consumption $P_t(pm)$ of a $pm \in PMs$ is modelled in [89] as a function of utilisation $util_t(pm)$ at time t, with P_{peak} and P_{idle} standing for the server's power consumption during peak and idle load, respectively.

$$P_t(pm) = pow(util_t, pm) = P_{idle} + util_t(pm) \cdot (P_{peak} - P_{idle}) \tag{5.5}$$

The impact of time series forecasting errors is modelled in [58], where the predicted value $\hat{x_t}$ of a real value x_t at time t is:

$$\hat{x}_t = \mathcal{N}(x_t, \sigma^2_{pred}),\ \forall t \in [t_0, t_N] \tag{5.6}$$

where $\mathcal{N}(x_t, \sigma^2_{pred})$ is a Gaussian distribution with mean x_t and standard deviation σ_{pred}.

Temperature-dependent cooling efficiency resulting from computer room air conditioning using outside air economizers is modelled in [180]. Cooling efficiency is expressed as partial PUE $pPUE_{dc,t}$ at data center dc at time t, which affects the power overhead based on the following formula:

$$pPUE_{dc,t} = \frac{P_t(pm) + P_{cool,t}(pm)}{P_t(pm)} = \frac{P_{tot,t}(pm)}{P_t(pm)} \tag{5.7}$$

where $P_{cool,t}(pm)$ is the power necessary to cool pm, and $P_{tot,t}(pm)$ stands for the combined cooling and computation power. The dynamic value of $pPUE_{dc,t}$ is modelled as a function of temperature T to match hardware specifics as:

$$pPUE_{dc,t} = 7.1705 \cdot 10^{-5} T^2_{dc,t} + 0.0041 T_{dc,t} + 1.0743 \tag{5.8}$$

Based on the migration model developed in [121], the combined energy consumption overhead of the source and destination hosts E_{mig} for a single migration can be calculated as a function of the migrated VM's memory V_{mem}, data transmission rate R, memory dirtying rate D and a pre-copying termination threshold V_{thd}.

$$E_{mig} = f(V_{mem}, D, R, V_{thd}) \tag{5.9}$$

Pervasive Cost Model

Based on our extensible resource types, we define a generic model of server utilisation $util_t(pm)$ at time t of a $pm \in PMs$ as a weighted sum of the individual resource type utilisations:

$$util_t(pm) = \sum_{\forall i \in \{1,\dots r\}} w_i \cdot \frac{\sum_{\forall vm \in alloc_t(pm)} res_{vm,i}}{cap_{pm,i}} \tag{5.10}$$

where w_i, $\forall i \in \{1,\dots r\}$ is a value in $[0,1]$ describing the weight resource type i has on the physical machine's power consumption (exact amounts depend on hardware specifics; the values we used are discussed in Section 5.4). Variables $cap_{pm,i}$ and $res_{vm,i}$ are the amounts of that resource available or requested by the *pm* or *vm*, respectively.

We model the power consumption $P_t(pm)$ of a $pm \in PMs$ using the basic approach from Eq. 5.5, but we extended it to model fast suspension of empty hosts (a technology explained in [140]). Also, to model additional load variation, we define P_{peak} and P_{idle} as time series of a server's power consumption during peak and idle load depending on the time t, instead of being constant.

$$P_t(pm) = \begin{cases} 0 & \text{if } util_t(pm) = 0 \\ pow(util_t, pm) & \text{otherwise.} \end{cases} \tag{5.11}$$

We use a common time series notation $\{x_t : t \in T\}$, where T is the index set and $\forall t \in T$, x_t is the time series value at time stamp t. For each data center location dc in DCs, there is a time series of real-time electricity prices $\{e_{dc,t} : t \in [t_0,t_N]\}$. Similarly, at each location there is a time series of temperature values $\{T_{dc,t} : t \in [t_0,t_N]\}$. To analyse forecasting errors, on both electricity and temperature time series, we apply Eq. 5.6. To explore its impact in the evaluation, we vary σ_{pred}, which determines the accuracy of the forecast. We assumed the temperature-dependent cooling efficiency model from Eq. 7.6 to express $P_{tot,t}$ and kept the polynomial model and the fitted factors from Eq. 7.7 where $pPUE$ ranges from 1.02 for -25 C to 1.3 for 35 C.

Combining all the equations so far, the cloud's energy cost C can be approximated using the rectangle integration method:

$$C = \sum_{pm \in PMs} \frac{t_N - t_0}{N} \sum_{t=t_0}^{t_N-1} P_{tot,t}(pm) e_{loc(pm),t} \tag{5.12}$$

Similarly, by omitting the electricity price component $e_{loc(pm),t}$ we calculate the cloud's energy consumption E. For adding the migration overhead, we considered the model from Eq. 5.9 and converted it to a cost using a mean electricity price between the locations at the time of the migration. Bandwidth costs were not considered, as the necessary business agreement details are not public – e.g. Google leases optical fiber cables, instead of paying for traffic. The final cost of all the migrations was added to the total energy consumption E and total energy cost C.

5.2.3 Optimisation Problem Specification

Based on the decision support components we can define the optimisation problem specification as $\{C_i : i \in \{1,\dots prob\_con\}\} \cup \{G_j : j \in \{1,\dots prob\_goal\}\}$, which are the sets of *prob_con* constraints and *prob_goal* optimisation goals composed of decision support components. This optimisation problem specification can be extended with arbitrary requirements. We now state the optimisation problem specification with two constraints and two goals that we use in our evaluation.

In every moment, every VM has to be allocated to one server (belong to its $alloc$ set) that acts as its host. This is the *allocation constraint* (C_1):

$$\forall t \in [t_0, t_N],\ \forall vm \in VMs_t,\ \exists_{=1} pm \in PMs,\ \text{s.t. } vm \in alloc_t(pm) \tag{5.13}$$

The *capacity constraint* (C_2) states that at any given time a server cannot host VMs that require more resources in sum than it can provide.

$$\sum_{\forall vm \in alloc_t(pm)} res_{vm,i} < cap_{pm,i},\ \forall pm \in PMs,\ \forall i \in \{1, \dots r\},\ \forall t \in [t_0, t_N] \tag{5.14}$$

The *cost goal* (G_1) is to minimise the cloud's electricity cost C expressed in Eq. 7.8, stemming from PM utilisation, cooling efficiency, electricity prices and migration overhead.

The *QoS goal* (G_2) is to minimise the rate of migrations $R_{mig}(vm)$ in a designated interval, $\forall vm \in VMs_t,\ \forall t \in [t_0, t_N]$. In the following section we present an optimisation engine for dealing with such a problem specification.

5.3 Forward-compatible Optimisation Engine

In this section we show concrete implementations of the optimisation engine workflow from Fig. 5.3. As already stated in Section 5.1.2, to tackle the NP-hard scheduling problem, we use a two-stage approach, with best-effort global optimisation and a deterministic local optimisation for hard constraint satisfaction. For the first stage global optimisation, we propose a genetic algorithm [87] where a population of potential solutions is evolved using genetic operators (crossover and mutation). For the second stage local optimisation, we propose a deterministic greedy local search where the best solution obtained by the genetic algorithim within the given time limit is further improved. The algorithm's main goal is to satisfy the hard capacity constraints, in case they were not already satisfied by the genetic algorithm, but it also considers the decision support components to reduce energy costs based on geotemporal inputs.

5.3.1 Algorithm Selection Justification

The reason the genetic algorithm was chosen for global optimisation (in the workflow from Fig. 5.3) is that using a fitness function for schedule selection matches the requirement of separated optimisation and solution evaluation logic. Furthermore, it satisfies the decision support component extensibility requirement through multiple fitness components with associated weights. There is also a benefit in keeping a population of solutions and not just a single best one, as is the case in deterministic optimisation techniques. Inputs change over time – requests to boot new or delete old VMs arrive, temperature or electricity price forecasts change. Upon such a change, our genetic algorithm propagates a part of the old population to the new environment and there is a higher chance that some solutions will still be fit (or a good evolution basis).

Greedy approaches are often used in deterministic local optimisation, e.g. in [44]. For the purpose of improving an existing schedule to satisfy primarily the hard constraints, without considering the full multi-objective trade-offs, it proved as a good addition to the genetic algorithm in our experiments.

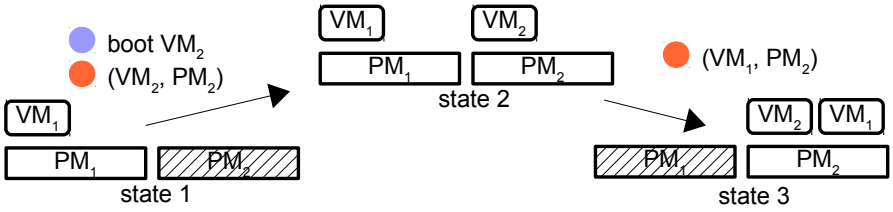


Figure 5.4: Example of state transitions based on control actions.

5.3.2 Forecast-based Planning

It is possible to forecast future values of geotemporal inputs to a certain extent [178, 19]. This facilitates planning of more efficient cloud management actions. For example, knowing whether a shift in electricity prices between two data center locations is the result of a temporary spike or a longer trend, enables more cost-efficient scheduling choices.

Time series forecasting is possible in a domain-agnostic manner, by dynamically fitting auto-regressive integrated moving average models [143]. As we are dealing with temperatures and electricity prices, widely used data, we assume domain-specific forecast information sources, such as the announcement of electricity prices (e.g. on day-ahead markets [178] and a weather forecast web service [19]).

5.3.3 Cloud Control Schedule

At the current moment t_c, we have information about future values for the geotemporal inputs for a period of time we call a *forecast window* that ends at t_f: $fw = [t_c, t_f]$. The size of the forecast window is determined by the available forecast data and the desired accuracy level. Given the current cloud configuration, we are able to estimate the effects of any cloud control actions in terms of the optimisation problem inside the forecast window by applying the cost model from Eq. 7.8. We represent a cloud control schedule as a time series $\{action_t : t \in fw\}$ of planned cloud control actions in the forecast window . In this chapter we consider VM live migration actions [121] and suspension of empty PMs that reduces idle power consumption [140]. A control action $action_t$ is described as an ordered pair (vm, pm), specifying which vm migrates to which pm. Migrations determine VM allocation over time (Eq. 5.4) and implicitly PM suspension (Eq. 5.11).

The representation of the cloud as a sequence of transitions between states (as defined through $alloc_t$ in Eq. 5.4), triggered by migration actions is illustrated in Fig. 5.4. Initially, vm_1 is hosted on PM_1. PM_2 is suspended, as it is empty. A migration of VM_2 to PM_2 transitions the cloud to a new state where PM_2 is awoken from suspension and hosting VM_2. An incoming request for VM booting can be represented as a migration with no source PM, like in the first transition. Next, an action migrates VM_1 to PM_2, after which PM_2 is hosting both VMs and PM_1 is suspended.

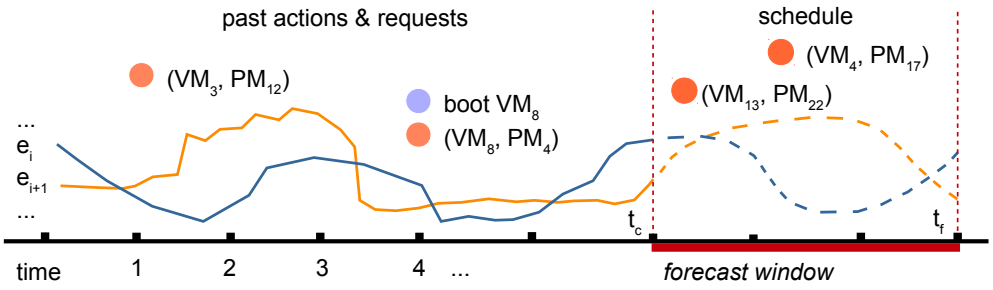


Figure 5.5: Schedule optimisation inside a forecast window.

The time-based aspect of a schedule is illustrated in Fig. 5.5. Past events that occurred before the current moment t_c, such as a VM migration at $time = 1$ or a new user request to boot a VM at $time = 4$, determine the current cloud state. Based on the past values of different geotemporal variables, such as electricity prices e_i and e_{i+1} at different data center locations, we are able to get their value forecasts in the forecast window. Different control actions of a schedule inside the forecast window can then be tried out at any moment between t_c and t_f. The control actions can be evaluated to determine the resulting VM locations and estimate costs with regard to the different geotemporal input forecasts and other optimisation problem aspects, such as constraints or SLA violations. When a schedule has been selected for execution, any immediate actions are applied and the forecast window moves as time passes, new requests arrive and geotemporal inputs change.

5.3.4 Hybrid Genetic Algorithm Implementation

We now present the hybrid genetic algorithm. The most challenging parts in its design were the genetic operators that had to semantically match our optimisation problem domain, the fitness function that combines the decision support components and the hybrid part of the algorithm, i.e. the greedy local improvement.

Schedule Fitness

In a genetic algorithm, we keep not only one, but a population of multiple problem solutions – multiple cloud control schedules, in our case. An essential part of the algorithm is to evaluate the fitness of each of these schedules. The fitness function we derived for this purpose is adapted from the decision support components, but constrained only to the forecast window (as we cannot affect actions that were already executed or plan further ahead than the available forecasts). Additional decision support components can be treated in a similar manner.

For a schedule s, we calculate the capacity and allocation constraint fitness, as a measure of how well the VMs are allocated and the PMs are within their capacity, using the time-weighted function:

$$constraint(s) = \sum_{\forall t \in fw} \frac{w_{alloc} \cdot R_{unalloc,t} + w_{cap} \cdot R_{overcap,t}}{|fw|} \tag{5.15}$$

where $R_{unalloc,t}$ and $R_{overcap,t}$ are ratios of $vm, \forall vm \in VMs_t$ for which C_1 (Eq. 5.13) does not hold and $pm, \forall pm \in PMs$ for which C_2 (Eq. 5.14) does not hold at moment t, respectively. $|fw|$ is the cardinality of fw (the number of time periods in the interval). Constants w_{alloc} and w_{cap} in $[0,1]$ define the components' weights.

The acceptable migration rate R_{mig_min} and the maximum allowed migration rate R_{mig_max} can be specified in the SLA, or determined by the cloud provider's bandwidth expenses. Their values are used to calculate the QoS component from the actual migration rate R_{mig} over the period of fw.

$$qos\_pen(vm) = \begin{cases} 0 & \text{if } R_{mig} < R_{mig\_min} \\ 1 & \text{if } R_{mig} > R_{mig\_max} \\ \frac{R_{mig} - R_{mig\_min}}{R_{mig\_max} - R_{mig\_min}} & \text{otherwise.} \end{cases} \tag{5.16}$$

The *qos_pen* expression gives a penalty for too frequent migrations per VMs that grows linearly from 0 to 1 for the migration rate R_{mig} from R_{mig_min} to R_{mig_max}. We then calculate the average QoS penalty over all the VMs.

$$qos(s) = \frac{\sum_{\forall vm \in VMs_{t_c}} qos\_pen(vm)}{|VMs_{t_c}|} \tag{5.17}$$

As a cost estimation, we use a simplified expression – $utilprice$. First, we calculate the average value of utilisation multiplied by $pPUE$ and e over all PMs in the forecast window. This is a heuristic of the total energy costs from Eq. 7.8.

$$up\_avg(s) = \frac{\sum_{\forall pm \in PMs} \sum_{\forall t \in fw} util_t(pm) \cdot pPUE_{loc(pm),t} \cdot e_{loc(pm),t}}{|PMs| \cdot |fw|} \tag{5.18}$$

Then we normalise it to a $[0,1]$ interval by dividing it with up_worst, which is the same as up_avg, but calculated for a constant maximum utilisation $util_t(pm) = 1$.

$$utilprice(s) = \frac{up\_avg(s)}{up\_worst} \tag{5.19}$$

To measure how tightly VMs are packed among the available PMs, we estimate the consolidation quality *consolid* by considering only positive utilisation time series elements denoted as $\{pos\_util_t(pm) : t \in fw, \text{ s.t. } util_t(pm) > 0\}$.

$$consolid(s) = 1 - \frac{1}{|PMs|} \sum_{\forall pm \in PMs} \frac{\sum_{\forall t \in fw} pos\_util_t(pm)}{|pos\_util(pm)|} \tag{5.20}$$

Finally, we can calculate the fitness as:

$$\begin{aligned} fitness(s) = w_{ct} \cdot constraint(s) + w_q \cdot qos(s) + \\ + w_{up} \cdot utilprice(s) + w_{cd} \cdot consolid(s) \end{aligned} \tag{5.21}$$

with w_{ct}, w_q, w_{up} and w_{cd} in $[0,1]$ determining each component's impact. The optimal schedule converges towards a fitness of 0 and the worst schedule towards 1. This solution evaluation form suits the forward compatibility requirement, as it can easily be extended with new decision support components by weighing them into the total fitness summation.

Genetic Operators

The *creation* procedure creates a random schedule as a time series of actions $(vm, pm),\ vm \in VMs_{t_c}, pm \in PMs$ at random times $t_r \in fw$. The number of migrations is uniformly distributed between the parameters *min_migrations* and *max_migrations*.

Given schedules s_1 and s_2, the *crossover* operator creates a child schedule s_3 by choosing a random moment $t_r \in [t_c, t_f]$:

$$s_3 = \left\{ \begin{array}{ll} s_{1,t}: & t \in [t_c, t_r], \\ s_{2,t}: & t \in [t_r, t_f] \end{array} \right\} \tag{5.22}$$

The *mutation* operator applied to a schedule s removes a random action a and inserts one at a random moment $t_r \in fw$:

$$s = s \setminus a \cup \{t_r : (vm, pm)\}, \tag{5.23}$$

$$a \in s, vm \in VMs_{t_c}, pm \in PMs \tag{5.24}$$

Algorithm

The core genetic algorithm (GA) is listed in Alg. 2. Among other parameters not introduced so far, it also receives the population size *pop*, crossover, mutation and random creation rates *cross*, *mut* and *rand* in $[0,1]$ and the maximum number of generations *gen*.

After creating the population, every schedule is evaluated using the fitness function (Eq. 5.21) in line 4. Schedules are selected for crossover using the *roulette wheel* selection method with chances for selection proportional to a schedule's fitness. A new generation is created by replacing the worst schedules with the newly created children (line 11). A random segment of the population proportional to *mut* is changed by applying the mutation operator (line 12). The termination condition is fulfilled after *gen* generations have passed (line 6) and the best schedule is returned. To cope with forecast-based planning, an existing population is partially propagated to the next schedule reevaluation, after the forecast window moves (line 1).

The GA is not guaranteed to satisfy all the constraints within a limited time period. To remedy this, we expand the optimisation with a greedy constraint satisfaction algorithm that is applied to the best schedule selected by the genetic algorithm to reallocate any offending VMs using a best cost fit (BCF) heuristic we developed that also considers geotemporal inputs.

Algorithm 2 Core genetic algorithm.

```
1:  schedules = create_population(pop, rand, [existing])
2:  i = 0
3:  while True do
4:      calculate_fitness(schedules)                          ▷ Eq. 5.21
5:      sort schedules by fitness, best first
6:      if i = gen then
7:          break loop
8:      end if
9:      parents = roulette_wheel(schedules, cross)
10:     children = crossover(parents)                         ▷ Eq. 5.22
11:     schedules = schedules[0 : pop · (1 − cross)] + children
12:     mutation(schedules, mut)                              ▷ Eq. 5.23
13:     i = i + 1
14: end while
15: existing = schedules
16: return schedules[0]
```

The greedy constraint satisfaction pseudo-code is listed in Alg. 3. The algorithm receives as input *schedule*, the output of the GA. We begin by marking for reallocation all VMs which cause any hard constraint violations (C_1 or C_2) in line 2 and additionally all VMs from underutilised hosts in line 3. The VMs will then be reallocated in the outermost loop (line 5) starting with larger VMs first whose placement is more constrained (line 4). Available PMs are split into *active* and *nonactive* lists, depending on whether they are suspended or not. We sort *inactive* in line 8 such that larger PMs come first for activation (preferable to more smaller machines, because of the idle power overhead) and data centers with lower combined electricity price and cooling overhead cost are preferred. The target PM to host *vm* is selected in the inner loop in line 10 by first sorting *active* to try and fill out almost full PMs first and prefer lower-cost location in case of ties (line 11) and activating the next PM from *inactive* if *vm* does not fit any of the *active* PMs (line 19). When a fitting PM is found, the action is added to *schedule* and the algorithm continues with the next VM.

5.4 Evaluation

We evaluated the proposed progressive cloud controller in a large-scale simulation of 10k VMs based on real traces of geotemporal inputs. To be able to simulate cloud behaviour under geotemporal inputs, we developed our own open source simulator Philharmonic. The goals of the evaluation are: (1) Estimate energy and cost savings, as well as the QoS attainable using the pervasive cloud controller; (2) Analyse the impact of various inputs, such as data center geography, different geotemporal inputs or controller parameters; (3) Validate the pervasive cloud controller extensibility by running the simulator with different decision support component subsets; (4)

Algorithm 3 Greedy constraint satisfaction – BCF heuristic.

```
1: to_alloc ← empty list
2: append all constraint-violating VMs to to_alloc
3: append VMs from all underutilised PMs to to_alloc
4: sort to_alloc by resource requirements decreasing
5: for vm ∈ to_alloc do
6:     active ← all PMs where at least one VM is allocated
7:     inactive ← all PMs where no VMs are allocated
8:     sort inactive by capacity decreasing, cost increasing
9:     mapped ← False
10:    while not mapped do
11:        sort active by free capacity, cost increasing
12:        for pm ∈ active do
13:            if vm fits pm then
14:                mapped ← True
15:                break loop
16:            end if
17:        end for
18:        if not mapped then
19:            pop inactive[0] and append it to active
20:        end if
21:    end while
22:    modify schedule by adding a migration (vm, pm)
23: end for
24: return schedule
```

Define cloud provider guidelines, such as how temperature variation or forecast data quality affect the energy savings and illustrate their usage in a case study.

5.4.1 Evaluation Methodology

In this part we give an overview of the simulator's implementation and proceed with explaining all the simulation details, such as the datasets, parameters and the baseline controller.

Philharmonic Simulator

A high-level overview of the Philharmonic simulator [72] that we developed is shown in Fig. 5.6. A simulation in Philharmonic consists of iterating through a series of equally-spaced time periods, collecting the currently available electricity price and temperature forecasts, as well as the incoming VM requests from the environment component. The controller is called with the data known at that moment about the environment and the cloud to reevaluate the schedule for the forecast window and potentially schedule new or different actions. The simulator applies any actions scheduled for the current moment on the cloud model and continues with the next time

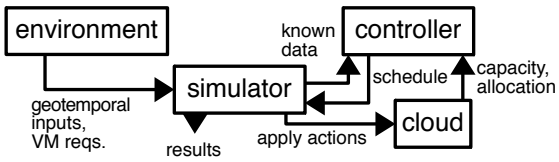


Figure 5.6: Philharmonic simulator overview.

Figure 5.7: Cities used as data center locations in the simulation based on a: (a) US dataset (b) world-wide dataset.

step, repeating the procedure. The applied actions are used to calculate the resulting energy consumption and electricity costs, using the model from Section 5.2.

Simulation Details

Real historical traces for electricity prices and temperatures were used in the simulation for 15 cities in the USA shown in Fig. 5.7 (a). The electricity price dataset described in [33] was used. The temperatures were obtained from the Forecast web service [19]. To evaluate less correlated geotemporal inputs, we used a world-wide dataset for six cities accross three continents shown in Fig. 5.7 (b), choosing non-US cities to match the locations of Google's data centers. Temperatures were again obtained from the Forecast API [19]. Due to lack of RTEP data for the four non-US cities, we artificially generated these traces from the data known for other US cities. We shifted

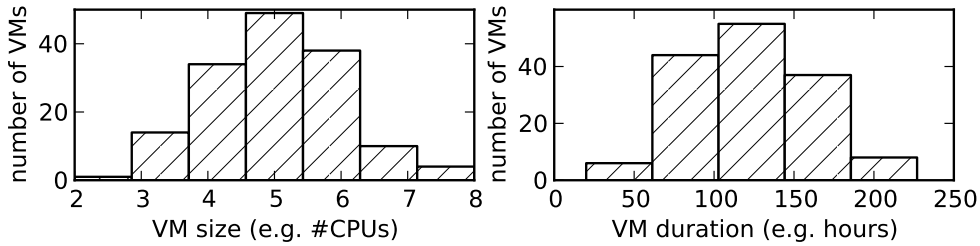


Figure 5.8: VM request resource and duration histogram.

the time series based on the time zone offsets and added a difference in annual mean values to resemble local electricity prices.

User requests for VMs were generated randomly by uniformly distributing the creation time and duration. The specifications of the requested VMs were modelled by normally distributing each resource type. An example VM request distribution is illustrated in Fig. 5.8. Available cloud infrastructure was generated by uniformly distributing physical machines among different data centers. Capacities for each each machine's resources were generated in the same manner as the VM requests.

period	duration	d	p (CPU; RAM)	v (CPU; RAM)	P_{peak}	P_{peak}
1 h	2 weeks	6	2,000 (8-16; 16-32)	10,000 (1-2; 2-4)	200	100
(μ_P, σ_P^2)	R_{mig_min}	R_{mig_max}	R	D	V_{thd}	
(0, 25)	1/4	1	1 Gb/s	0.3 Gb/s	0.1 Gb/s	

Table 5.2: Simulation parameters

The exact simulation parameters used in the evaluation are listed in Table 5.2. The time is defined by its period (simulation step size) and the total duration that determines the number of steps. To define the cloud, the number of data centers is given as d (for the world-wide scenario, for the US scenario we consider $d = 15$), and for PMs and VMs their number p and v (of boot requests in case of VMs – there were about 50% as much delete events as well) with minimum and maximum resource values in parentheses for the resources we assumed in this simulation – number of CPUs and the amount of RAM in GB. Besides running the simulation for the large-scale scenario with 10k VMs, we also simulated 100 and 1k VMs, but as the difference in normalised savings was only marginal, we only include the large-scale results. Uniform resource weights were used in the $util$ function (Eq. 5.10). We used P_{peak} with P_{idle} values 50% of the peak, as reported in [77], and added normal random noise of the form $\mathcal{N}(\mu_P, \sigma_P^2)$ to account for load variation. R_{mig} was calculated hourly and constant migration model parameters (R, D, V_{thd}) were used. We used these settings in all the simulations, unless otherwise specified.

As a baseline controller for results comparison, we implemented a method for VM consolidation dynamically adapting to user requests using a best fit decreasing (BFD) placement heuristic

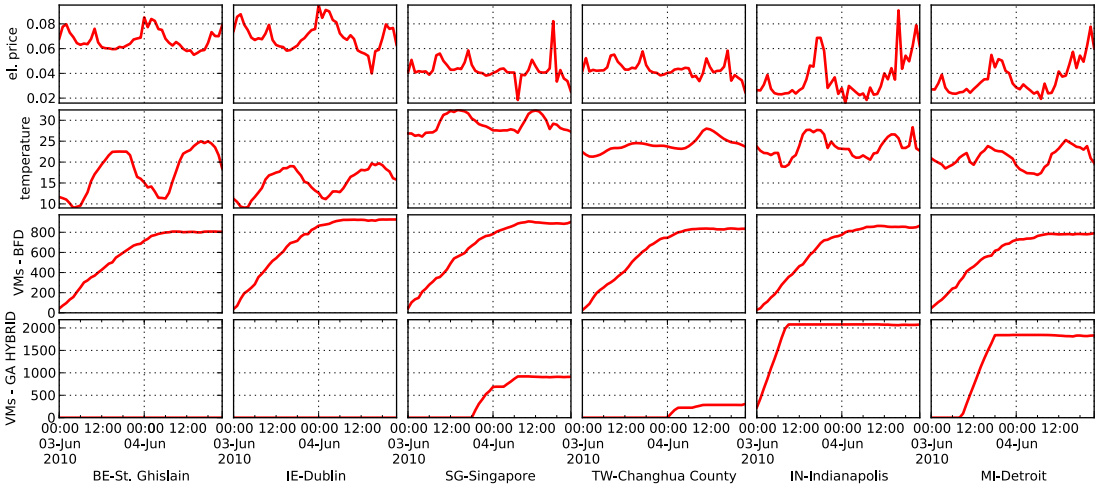


Figure 5.9: Dynamic VM management comparison of the controllers.

developed by Beloglazov et al. in [44]. We implemented the updated version of the controller that is currently developed for inclusion in the OpenStack open source cloud manager as project Neat [45]. Based on our classification of related work, this is a level two cloud management method that dynamically reallocates VMs, treating all energy uniformly.

fw	pop	gen	cross	mut	rand	*min_migr*	*max_migr*
12 h	100	100	0.15	0.05	0.3	0	$fw\|VMs_{t_c}\|/3$
w_{up}	w_{ct}	w_{alloc}	w_{cap}	w_q	w_{cd}		
0.4	0.1	0.4	0.6	0.4	0.1		

Table 5.3: Optimisation engine parameters

The optimisation engine's algorithm parameters are listed in Table 5.3. All the weights used in the fitness function (Eq. 5.21) were systematically calibrated using automated parameter exploration, which we cover later. The values listed in the table show the parameter combination that achieved the highest energy savings with the least number of constraint violations.

5.4.2 Dynamic Controller Analysis

This part of the analysis aims to compare our pervasive cloud controller with the baseline controller by visualising individual actions. The use case is the scenario with world-wide data centers and other parameters we already described in Table 5.2.

In Fig. 5.9 we see 6x4 graphs, where the columns represent data centers. First two rows show geotemporal inputs – electricity price and temperatures. The next two row shows the dynamic number of active VMs at the corresponding data center. The third row shows the behaviour of

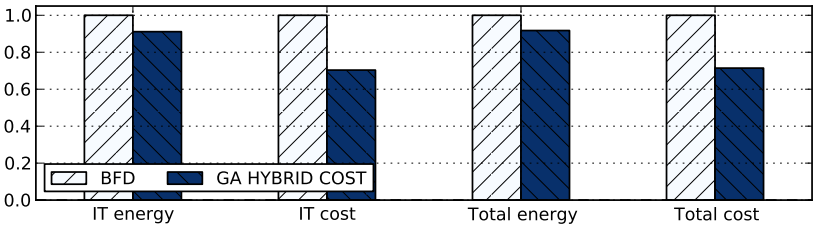


Figure 5.10: Normalised energy and costs of the pervasive cloud controller compared to the baseline method in a simulation of 10k VMs.

the baseline controller and the fourth row of the pervasive cloud controller. The x-axes of all the graphs cover the same time span and the graphs in the same column are aligned to the same x-axis, shown in the bottom. Similarly, the graphs in the same row share the same y-axis.

The electricity prices are lowest in the USA (with an increase towards the end of the day), followed by Asian locations and the European locations have significantly higher prices. Temperatures start the lowest in Europe, but then approach 20 C. The other locations oscillate around 20 C, except for the peaks in Asian locations where 30 C are approached. We can see that the baseline method roughly uniformly distributes the VMs across all the available data centers, disregarding the geotemporal inputs. The pervasive cloud controller allocates VMs in the first 18 hours filling out the US capacities, targeting lower electricity prices. No VMs are allocated in the European locations during this period, due to high electricity prices and enough capacity at other locations. The Asian locations are initially empty, but after the temperature peak is over, VMs are migrated to the Singapore data center and at the end of the first day (when Asian electricity prices start to decrease even bellow the US values), the Taiwan data center as well. This shows us the desired behaviour of the pervasive cloud controller where geotemporal inputs are monitored and adapted to by reallocating load to the most cost-efficient data center location.

5.4.3 Aggregated Simulation Results

To give an estimation of the benefits of using our pervasive cloud controller in a large-scale scenario described in Table 5.2 and analyse various environmental parameters, we collected the performed actions during the whole simulation and calculated the aggregated energy consumption and costs.

Cost Savings

The normalised results of the simulation based on the world-wide dataset with 10k VMs are shown in Fig. 5.10. A group of columns is shown for each of the examined quality metrics – IT energy, IT cost, total energy and total cost. Inside each group, there is a column for both of the scheduling algorithms: the baseline algorithm (BFD) and the pervasive cloud controller (GA HYBRID COST). The values are normalised as a relative value of the baseline algorithm's results. The absolute values are listed in Table 5.4. The pervasive cloud controller achieves savings of 28.6% in total energy cost compared to the baseline. We can see that significant savings can

	BFD	GA HYBRID COST
IT energy (kWh)	6226.00	5673.63
IT cost ($)	309.40	217.64
Total energy (kWh)	7488.01	6869.55
Total cost ($)	370.20	264.39

Table 5.4: Absolute energy consumption and costs

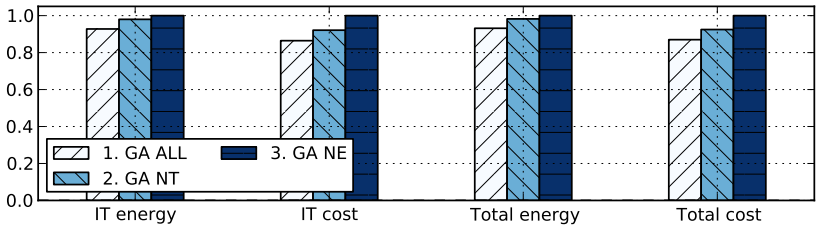


Figure 5.11: Normalised energy and costs of the pervasive cloud controller with various decision support components: temperature and electricity price (1), no temperature (2) and no temperature or electricity cost (3).

be achieved using our pervasive cloud controller, which is especially relevant for large cloud providers such as Google or Microsoft that spend over $40M annually on data center electricity costs [155].

Decision Support Component Variation

To validate the controller's extensibility and show that it can work with different decision support components, we performed the same 10k VM simulation with different subsets of the decision support components considered by the optimisation engine. This analysis also gives an overview of the impact individual geotemporal inputs have in the total achieved energy savings.

The results are shown in Fig. 5.11. Each column stands for one of the simulation scenarios – both temperatures and electricity price components (GA ALL), electricity price component, but no temperature (GA NT) and no electricity price or temperature components (GA NE). Total cost savings of 7.5% are achieved when both components are considered compared to not considering temperatures. Savings are 13% when compared to not considering both components, which is a significant difference. In the same manner we turn on or off certain decision support components as a configuration option in the controller's implementation, new geotemporal inputs and rules can be added in the future when necessary.

Geography Variation

Different cloud providers will have different data center locations and geographical distributions. To estimate the impact of this geographical distribution of data centers on the possible cost savings achievable using our scheduling method, we simulate its effects on two such scenarios –

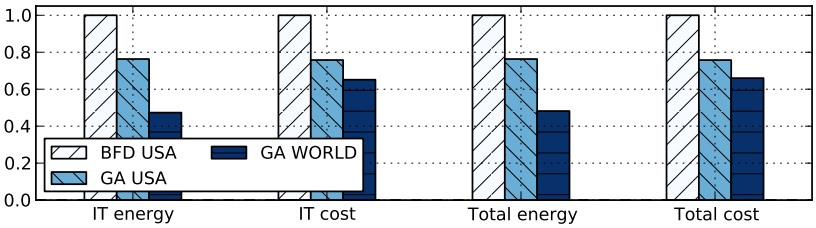


Figure 5.12: Normalised energy and costs of the pervasive cloud controller compared to the baseline method for the USA and world-wide datasets.

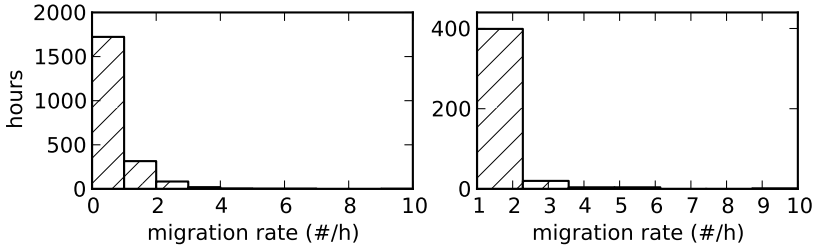


Figure 5.13: Hourly migration rate histogram (two zoom levels).

the world-wide scenario and the USA scenario, as described in Section 5.4.1. Furthermore, as the USA dataset of geotemporal inputs consists of real historical electricity price traces (which we did not have to artificially adapt to different time zones and local averages) it further testifies to the validity of our approach. Lastly, even though current cloud providers have incentives to spread their data centers further apart to bring services closer to a world-wide user base, with the advent of smart buildings [153] we might see more localised data center distributions based on neighborhood, city or region organisations.

The results of the simulation for the USA and world-wide dataset for 10k VMs are shown in Fig. 5.12. The simulation settings were the same we explained in Section 5.4.1, except for the locations of the physical machines. The baseline controller was run for the USA dataset (BFD USA), and we can see the normalised results compared to this baseline for the pervasive cloud controller simulated on the USA dataset (GA USA) and the world-wide dataset (GA WORLD). It can be seen that significant cost savings of 24% are achieved even for the USA-only scenario. The pervasive cloud controller's energy consumption and costs are lower in the world-wide scenario than for the US-only data centers, though – a further 10% decrease is possible.

QoS Analysis

Aside from understanding the cost savings from the cloud provider's perspective, we also have to analyse the QoS, i.e. how the controller affects end users of VMs. To measure this, we count how often the migration actions occur, i.e. the migration rate. To get more data, we ran the simulation to cover three months. A histogram of hourly migration rates of all the VMs obtained from the simulation of the pervasive cloud controller can be seen in Fig. 5.13. The two plots show different

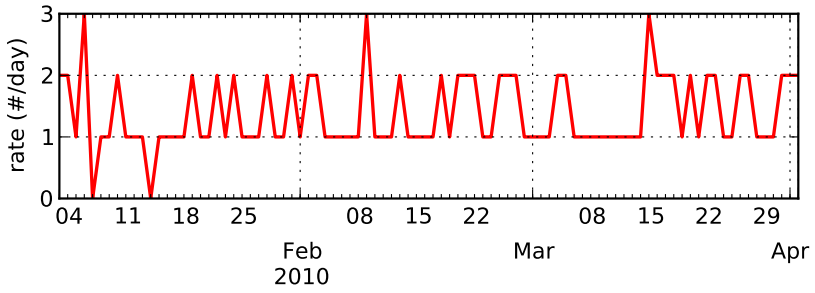


Figure 5.14: Aggregated worst-case per-VM migration rate.

zoom levels, as there are progressively less hours with higher migration rates. Most of the time, no migrations are scheduled, with one migration per hour happening about 20% of the time.

To get a QoS metric meaningful to the user, we group migrations per VM (as users are only interested in migrations of their own VMs) and process them in a function that aggregates migrations over a daily interval. We then define the *aggregated worst-case* metric by counting the migrations per VM per day and selecting the highest migration count among all the VMs in every interval. Such a metric could be useful e.g. in defining the lower bound for the availability rate in an SLA. The aggregated worst-case migration rate for the simulation of the pervasive cloud controller is shown in Fig. 5.14. There are one or two migrations per VM per day most of the time, with an occasional case with a higher rate such as the peaks with three migrations.

Given that this data is highly dependent of the scheduling algorithm parameters used and the actual environmental parameters for a specific cloud deployment, fitting one specific statistical distribution to the data to get the desired percentile value that can be guaranteed in an SLA would be hard to generalise for different use cases and might require manual modelling. Instead, we propose applying the distribution-independent bootstrap confidence interval method [74] to estimate the aggregated migration rate. In our simulation, the 95% confidence interval for the mean daily per-VM migration rate is 1.26–1.5 migrations per day.

Genetic Algorithm Parameter Exploration

To explore how the optimisation engine behaves under different GA parameters, we ran the simulation with different parameter values and compared the resulting energy costs. We explored the weights of the different fitness function components in (Eq. 5.21). We developed a method for automatically running the simulation with different parameter combinations in the Philharmonic simulator. We covered a set $\{0, 0.1, 0.2, \dots\ 1.\}$ for each of the four weight parameters, exploring all the combinations with a constraint that their sum equals 1 (as only weight ratios make a difference in the GA, not their absolute values), resulting in 285 combinations. This method can be used to calibrate the controller for different environments by finding the parameter combination that achieves the highest energy savings or the best QoS, similar to how different objectives are optimised in a Pareto frontier.

A radio chart with the results is shown in Fig. 5.15. The five axes show the four fitness component weights with the fifth axis as the energy cost expressed relative to the worst-case

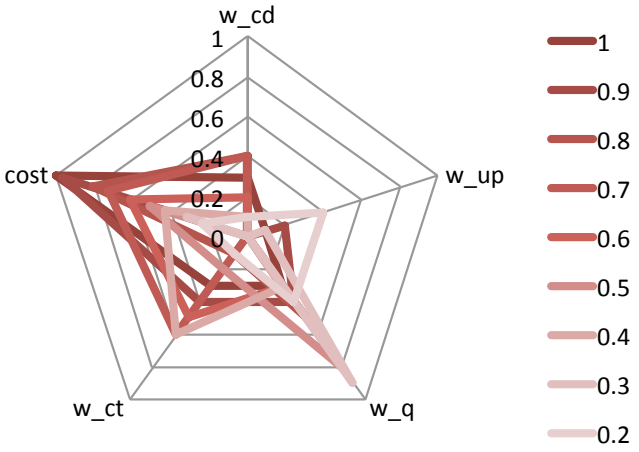


Figure 5.15: Relative energy costs for different GA parameters.

combination. One combination is shown as a pentagon of the same colour, indicating the 5-tuple $(w_{ct}, w_q, w_{cd}, w_{up}, cost)$. Combination colour is sorted by relative cost as well, with darker colours having higher and lighter colours lower energy costs. For clarity, we only show a subset of the combinations with the rounded relative cost closest to a step of 0.1. We can see that the lowest energy costs are achieved for high w_{up} and w_q weights, meaning that energy cost and a low number of migrations is prioritized. Higher energy costs were obtained when constraint satisfaction (w_{ct}) and VM consolidation (w_{cd}) is prioritized, as neither of these components includes geotemporal inputs.

Temperature Range Variation

As different cloud providers have data centers at various locations, where temperature ranges can be very different, we analyse the impact of temperature range variation on pervasive cloud control effectiveness. Temperature variation is affected by the time of the year and the range of daily temperatures will vary over time. For this reason, we performed the simulation with different starting times throughout the year, which resulted in different temperature ranges for different simulation runs.

The resulting graphs are shown in Fig. 5.16. The figure to the left shows the energy cost (normalised as relative to the maximum vaule) resulting from the simulation of the pervasive cloud controller handling the same VM requests, only shifted to a different month of the year. We can see a gradual trend, with a single sudden drop in October. To extract the statistical environment changes between these runs, we plotted the same normalised cost as a scatter plot against the temperature variation in the figure to the right. The temperature variation is calculated as the mean of the standard deviations of temperature values for individual data center locations. Once ordered this way, the trend of the data becomes clearer and we calculated a linear correlation between the variables with an adjusted R^2 of 0.31 (model shown in red). We see that a higher

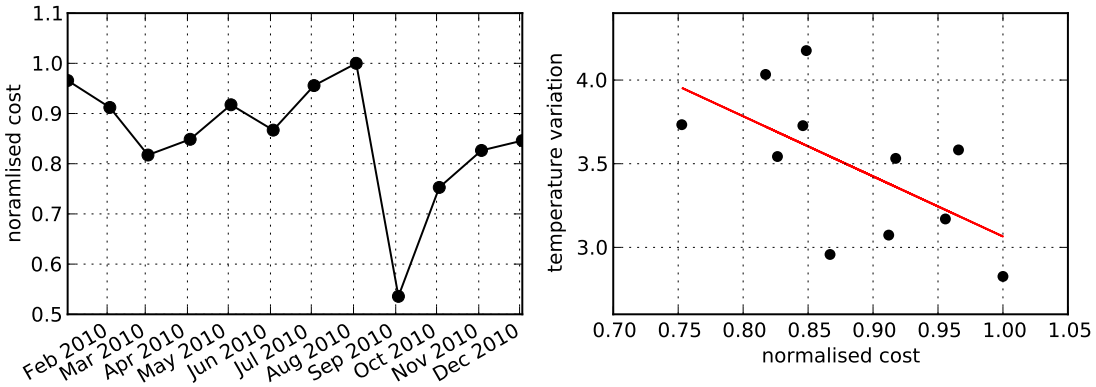


Figure 5.16: Energy costs for simulation runs during different months of the year (left) and a scatter plot of the same data and the temperature variation metric showing linear dependence (right).

temperature variation results in lower energy costs. This is due to the higher impact avoiding locations with unfavourable cooling efficiency conditions has.

Data Quality

To explore the effect errors in forecasting geotemporal inputs have on the pervasive cloud controller's operation, we simulate different data quality scenarios. Additionally, we explore different forecast window sizes and their effect on scheduling efficiency. A longer forecast window enables the fitness function to evaluate the consequences of different management actions over a longer interval, reducing the impact of short-term geotemporal impact changes, such as electricity price spikes.

The time series provided to the scheduling algorithm with different forecasting errors were obtained using Eq. 5.6 by selecting different standard deviation (σ_{pred}) parameters. A σ_{pred} close to zero represents very accurate forecasting, while a higher σ_{pred} causes higher signal volatility and forecasting errors. A segment of the generated time series for one of the cities is shown in Fig. 5.17. Both time series are aligned to the same x-axis. Each curve represents one σ_{pred} scenario. It can be seen that smaller error levels (σ_{pred} of 3 \$/MWh and 0.5 C for electricity or temperature, respectively) still retain the general trend with identifiable peaks and lows. Higher error levels (σ_{pred} from 30 to 50 \$/MWh or 3 to 5 C for electricity or temperature, respectively) start to significantly diverge from the original time series, in that peaks are predicted where in reality lows occur and vice versa. We simulated forecast window sizes of 4, 12, 24 and 48 hours.

The generated time series with forecasting errors were provided to the pervasive cloud controller to base its decisions upon. The forecast parameter exploration results are shown in Fig. 5.18. The 3-dimensional visualisation shows the space of forecast window sizes and electricity price σ_{pred} values on the bottom plane. The z-position on the surface (its height) shows the total energy cost (including the migration and cooling overhead), normalised relative to the

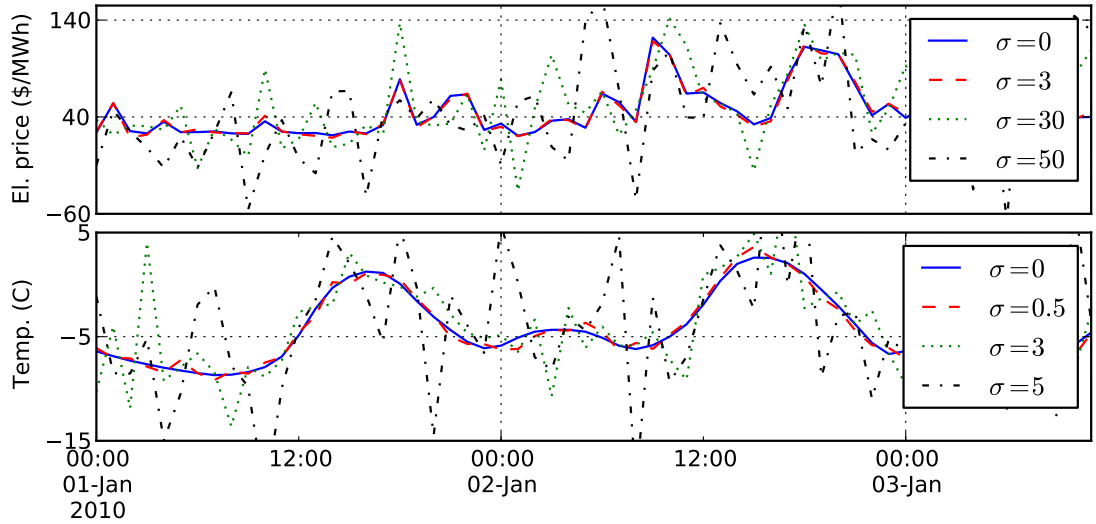


Figure 5.17: Different levels of forecasting errors applied to electricity price and temperature time series (for Mankato, MN).

worst case ($fw = 4\ h,\ \sigma_{pred} = 50\ \$/MWh$). Missing data points were interpolated. The graph only shows σ_{pred} used to model electricity price forecasting errors, but a matching σ_{pred} for temperature from Fig. 5.17 was also applied.

Looking at the forecast window sizes in Fig. 5.18, we can see that initially bigger windows result in lower costs. This trend can clearly be seen from 4 h to 20 h for all σ_{pred}. The trend changes, however, for 24 h and bigger windows. Higher or lower savings are visible and the pattern is more randomised. The reason is that both electricity prices and temperatures exhibit a daily seasonality effect and extending the window further than 24 h does not provide much more information to the controller, but increases the problem search space. We conclude that the rift-like surface shape in the forecast window range from 12 to 24 hours represents an optimal size for the given geotemporal inputs.

The forecasting error dimension shows deviations of around 25% between large forecasting errors and perfect knowledge. This can be attributed to the fact that large forecasting errors mislead the controller into placing VMs in areas where geotemporal inputs are in fact worse, so both energy cost losses and migration overheads are incurred. Smaller forecasting errors result in lower energy costs, which shows the importance of having accurate forecasting methods (or data sources) when managing clouds based on geotemporal inputs. Based on our simulation, σ_{pred} of 3 $/MWh (mean squared error (MSE) of ≈ 9) and 0.5 C (MSE of ≈ 0.25) or less is necessary for feasible cost savings.

5.4.4 Cloud Provider Guidelines Case Study

To show the usage potential of the collected measurements as guidelines, we performed a case study for several different cloud providers. The results are shown in Table 5.5. The annual electricity cost estimations are obtained from [155]. We selected cloud providers of different scale (e.g. A and C). We compare several environment conditions, namely the temperature

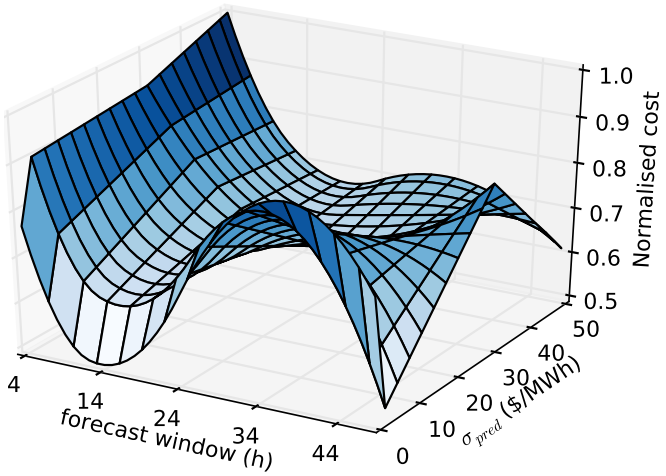


Figure 5.18: Normalised energy costs resulting from different forecasting errors and forecast window sizes.

variation *temp_var* and data quality metrics fw and σ_{pred}, considering different combinations as hypothetical scenarios that cloud providers might be in. The cost factor is calculated based on the already analysed temperature variation linear model from Fig. 5.16 and the forecast data quality impact results from Fig. 5.18. Finally, we show the order of magnitude of the cost savings based on the cost factor.

provider	electricity	fw	σ_{pred}	*temp_var*	factor	savings
A	$38M	14	10	3.5	0.565	$16.5M
B	$36M	48	30	2.5	0.989	$0.4M
C	$12M	24	10	4.0	0.536	$5.6M

Table 5.5: Cloud provider guidelines case study.

The results show that significant savings are possible using pervasive cloud control with appropriate environmental conditions. The B scenario, however, shows that even with high initial costs, having bad forecast data quality and a low temperature variation can result in lower savings, perhaps not enough of an incentive to apply our method. On the other hand, even a smaller provider, such as C, can achieve promising savings in a favourable environment.

5.5 Summary

In this chapter we presented an approach for pervasive cloud control under geotemporal inputs, such as real-time electricity pricing and temperature-dependent cooling efficiency. The solution is designed for extensibility with new geotemporal inputs and cloud regulation mechanisms through a modular decision support component system and a forward-compatible optimisation engine.

We presented a proof-of-concept controller implementation combining forecast-based planning and a hybrid genetic algorithm with greedy local optimisation. The genetic algorithm approach was extended with partial population propagation.

The approach was evaluated in a simulation based on real traces of temperatures and electricity prices. We estimated energy cost savings of up to 28.6% compared to a baseline cloud control method that applies VM consolidation without considering geotemporal inputs. We analysed per-VM migrations to show that no significant QoS impact is incurred in the process. We evaluated different parameters such as geographical data center distributions and forecast data quality as cloud provider guidelines to find conditions fit for pervasive cloud control.

CHAPTER 6

Progressive SLA Specification for Energy-Aware Cloud Management

The pervasive cloud controller from the previous chapter dynamically reallocates computation across geographically distributed data centers to leverage regional electricity price and temperature differences. As a result, a managed VM may suffer occasional downtimes. Current cloud providers only offer high availability VMs, without enough flexibility to apply such energy-aware management. In this chapter, we show how to analyse past traces of dynamic cloud management actions based on electricity prices and temperatures to estimate VM availability and price values. We propose a novel SLA specification approach for offering VMs with different availability and price values guaranteed over multiple SLAs to enable flexible energy-aware cloud management. We determine the optimal number of such SLAs as well as their availability and price guaranteed values. We evaluate our approach in a user SLA selection simulation using Wikipedia and Grid'5000 workloads. The results show higher customer conversion and 39% average energy savings per VM.

Section 6.1 gives an overview of how our progressive SLA specification can be applied in energy-aware cloud management based on geotemporal inputs. The probabilistic technique to build SLAs is presented in Section 6.2. The user behaviour model is described in Section 6.3. The evaluation description along with the results analysis are given in Section 6.4.

6.1 Progressive SLA Specification

To be able to reason about energy-aware cloud management in terms of SLA specification, we analyse two concrete schedulers: (1) The migration scheduler described in Chapter 5 – applies a genetic algorithm to dynamically migrate VMs, such that energy costs based on geotemporal inputs are minimised, while also minimising the number of migrations per VM to retain high availability. (2) The peak pauser scheduler described in Chapter 4 – pauses the managed VMs

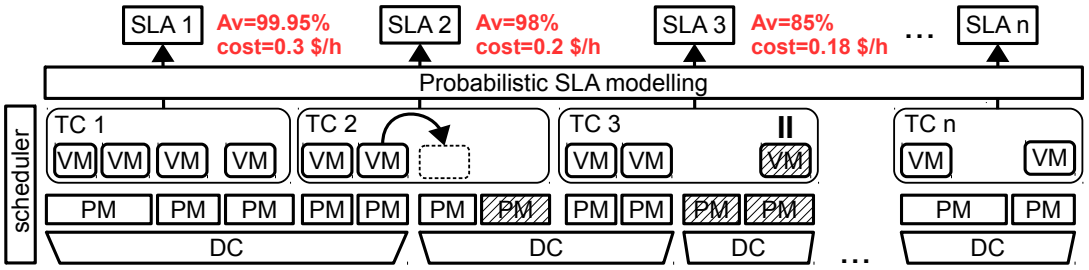


Figure 6.1: Progressive SLA specification

for a predefined duration every day, choosing the hours of the day that are statistically most likely to have the highest energy cost, thus reducing VM availability, but also the average energy cost.

With such energy-aware cloud management methods in mind, we propose a progressive SLA specification, where services are divided among multiple treatment categories, each under a different SLA with different availability and price values. Hence, different schedulers can be used for VMs in different treatment categories (or the same scheduler with different QoS constraint parameters). The goal of this approach is to allow different levels of energy-aware cloud management and thus achieve higher energy savings on VMs with lower availability requirements. What is given, therefore, are the schedulers for each treatment category, and the historical traces of generated schedules. What we have to find are the availability and price values that can be guaranteed in the SLAs for each treatment category and the optimal number of such SLAs to balance SLA flexibility and search difficulty for users.

We illustrate this approach in Fig. 6.1. A cloud provider operates a number of VMs, each hosted on a PM located in one of the geographically distributed data centers (DC). Each VM belongs to a treatment category (TC) that determines the type of scheduling that can be applied to it. For example, TC 1 is a high availability category where no actions are applied on running VMs. TC 2 is a category for moderate cloud management actions, such as live VM migrations (marked by an arrow) which result in short downtimes. TC 3 is a more aggressive category where VMs can be paused (marked as hatched with two vertical lines above it) for longer downtimes. Other TCs can be defined using other scheduling algorithms or by varying parameters, e.g. the maximum pause duration. The optimal number of TCs (and therefore also SLAs) n is determined by analysing user SLA selection to have enough variety to satisfy most user types, yet not make the search too difficult, which we explore in Section 6.4. Aside from selecting the number of SLAs, another task is setting availability and price values for every SLA. As energy-aware cloud management that depends on geotemporal inputs introduces a degree of randomness into the resulting availability and price values of a VM, we can only estimate the values that can be guaranteed. We do this using a *probabilistic SLA modelling* method for analysing historical cloud management action traces to calculate the most likely worst-case availability and average energy cost for a VM in a TC. For the example in Fig. 6.1, sample values are given for the SLAs. SLA 1 might have an availability (Av) of 99.95% and a high cost of 0.3 $/h due to no energy-aware management, SLA 2 might have slightly lower values due to live migrations being applied on VMs in TC 2, SLA 3 might have even lower values due to longer downtimes caused by VM

pausing etc. In the following section, we will show how to actually estimate availability and price values for the SLAs using probabilistic SLA modelling.

6.2 Probabilistic SLA Modelling

To estimate availability and price values that can be guaranteed in an SLA using probabilistic modelling for a certain TC, we require historical cloud management traces. Cloud management traces can be obtained through monitoring, but for evaluation purposes we simulate different scheduling algorithm behaviour. VM price is estimated by accounting for the average energy costs. To calculate VM availability, we analyse the factors that cause VM downtime. While the downtime duration of the peak pauser scheduler can be specified beforehand, the total downtime caused by the migration scheduler depends on VM migration duration and rate as dictated by geotemporal inputs, so we individually analyse both factors.

6.2.1 Cloud Management Simulation

Our modelling method can be applied to different scheduling algorithms and cloud environments. To generate a concrete SLA offering for evaluation purposes, we consider a use case of a cloud consisting of six geographically distributed data centers shown in Fig. 6.2 to represent a deployment similar to large cloud providers such as Google. We use the world-wide datasets of electricity prices [33] and temperatures from the Forecast web service [19] described in Section 5.4.1. The effects of the migration and peak pauser scheduler are determined in a simulation using the Philharmonic cloud simulator also described in Section 5.4.1. The cloud simulation parameters are summarised in Table 6.1. We illustrate the application of the presented methods with this use case as a running example.

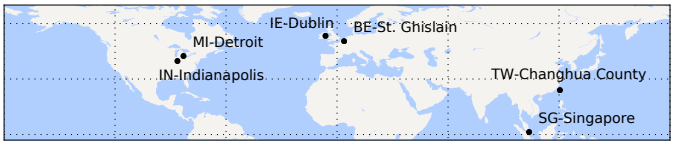


Figure 6.2: Simulated world-wide data centers

duration	DCs	PMs	VMs
3 months	6	20	80

Table 6.1: Cloud simulation

6.2.2 Migration Duration

Even a live VM migration incurs a temporary downtime, in the stop-and-copy phase of VM memory transferring. A very accurate model, with less than 7% estimation errors, for calculating this downtime overhead is presented in [121]. The total VM downtime during a single live migration T_{down} is a function of the VM's memory V_{mem}, data transmission rate R, memory dirtying rate D, pre-copying termination threshold V_{thd} and T_{resume}, the time necessary to resume a VM.

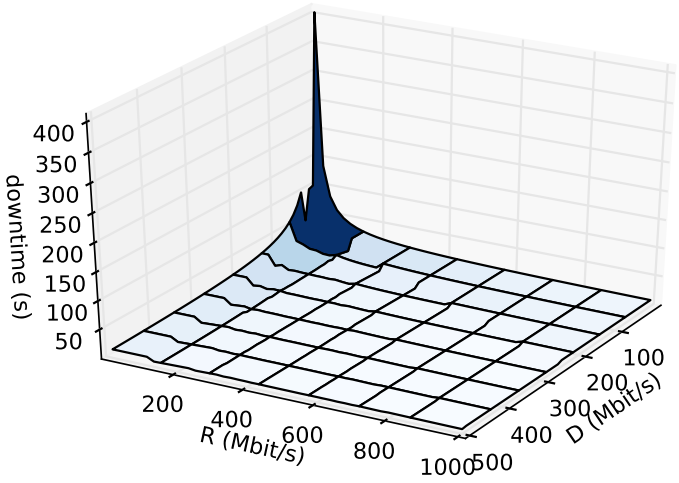


Figure 6.3: VM migration downtime

$$T_{down} = \frac{V_{mem} D^n}{R^{n+1}} + T_{resume} \quad \text{, where} \quad n = \left\lceil log_{\frac{D}{R}} \frac{V_{thd}}{V_{mem}} \right\rceil \tag{6.1}$$

All of the parameters can be determined beforehand by the cloud provider, except for R and D which depend on the dynamic network conditions and application-specific characteristics. Based on historical data, it is possible to reason about the range of these variables. In our running example, we assume a historical range from low to high values. R values in the 10–1000 Mbit/s range were taken based on an independent benchmark of Amazon EC2 instance bandwidths. D values from 1 kbit/s (to represent almost no memory dirtying) to 1 Gbit/s were taken (the maximum is not important, as will be shown). We assumed constant $V_{mem} = 4GB$, $V_{thd} = 1GB$, $T_{resume} = 5s$, as these values do not affect the order of magnitude of T_{down}. We show how T_{down} changes for different R and D in Fig. 6.3. Looking at the graph, we can see that higher R and D values result in convergence towards negligible downtime durations and the only area of concern is the peak happening when R and D are both very low. This happens, because close R and D values lead to a small number of pre-copying rounds and copying the whole VM under slow speeds leads to long downtimes. T_{down} is under 400 s in the worst-case scenario, which will be our SLA estimation as suggested in [48].

6.2.3 Migration Rate

Aside from understanding migration effects, we need to analyse how often they occur, i.e. the migration rate. We presented a method to analyse migration traces obtained from the cloud manager's past operation in Chapter 5. A histogram of migration rates for the migration traces from our running example described in Section 6.2.1 can be seen in Fig. 6.4. We reuse the *aggregated worst-case* function presented in Section 5.4.3 that counts the migrations per VM per

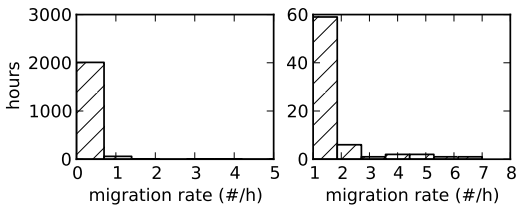


Figure 6.4: Hourly migration rate histogram

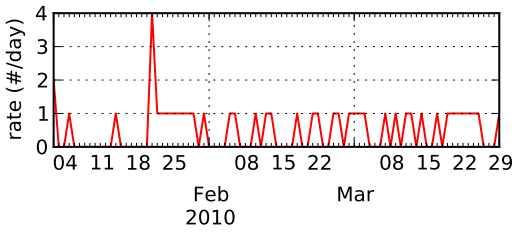


Figure 6.5: Aggregated worst-case migr. rate

day and selects the highest migration count among all the VMs in every interval. The output time series is shown in Fig. 6.5. There is one or zero migrations per VM most of the time, with an occasional case with a higher rate, such as the peak in January. Such peaks can occur due to more turbulent geotemporal input changes. or insufficient schedule optimisation.

Finally, the bootstrap confidence interval method [74] was applied to predict the maximum aggregated migration rate. For our migration dataset, the 95% confidence interval for the worst-case migration rate is from three to four migrations per day.

6.2.4 SLA Options

By combining the migration rate and duration analyses, we can estimate the upper bound for the total VM downtime and, therefore, the availability that can be warranted in the SLA. We define availability (Av) of a VM as:

$$Av = 1 - \frac{\text{total VM downtime}}{\text{total VM lease time}} \tag{6.2}$$

For our migration dataset and the previously discussed migration duration and rate, we estimate the total downtime of a VM controlled by the migration scheduler to be 27 minutes per day in the worst case, meaning we can guarantee an availability of 98.12%. We can precisely control the availability of the VMs managed by the peak pauser.

The average energy savings ($en\_savings$) for a VM running in a treatment category TC_i can be calculated by comparing it to the high availability TC_1. From the already described simulation, we calculate en_cost, the average cost of energy consumed by a VM based on real-time electricity prices and temperatures. We divide the energy costs equally among VMs within a TC. This is an approximation, but serves as an estimation of the energy saving differences between TCs. We calculate energy savings as:

$$en\_savings(VM_{TC_i}) = 1 - \frac{en\_cost(VM_{TC_i})}{en\_cost(VM_{TC_1})} \tag{6.3}$$

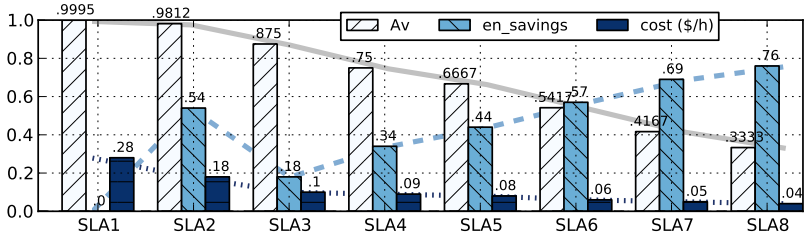
where $en\_cost(VM_{TC_1})$ is the average energy cost for a VM in TC_1 with no actions applied and $en\_cost(VM_{TC_i})$ is the average energy cost for a VM in the target TC_i.

The VM cost consists of several components. Aside from en_cost, $service\_cost$ groups other VM upkeep costs (manpower, hardware amortization, profit margin etc.) during a charge unit

(typically one hour in IaaS clouds). We assume the service component to be charged linearly to the VM's availability.

$$cost = en\_cost + Av \cdot service\_cost \tag{6.4}$$

To generate the complete SLA offering we consider an Amazon m3.xlarge instance which costs 0.280 $/h (Table 6.2) as a base VM with no energy-aware scheduling. Base instances with different resource values (e.g. RAM, number of cores) can be used, but this is orthogonal to the QoS requirements of availability that we consider and would not influence the energy-aware cloud management potential. Similarly, Amazon spot instances were not considered specially as they perform exactly the same as normal instances while running, as we mentioned in Chapter 2. We assumed the service component to be 0.1 $/h, about a third of the VM's price. The prices of VMs controlled by the two energy-efficient schedulers were derived from it, applying *en_savings* obtained in the cloud simulation. The resulting SLAs are shown in Fig. 6.6. SLA 1 is the base VM. SLA 2 is the VM controlled by the migration scheduler. The remaining SLAs are VMs controlled by the peak pauser scheduler with downtimes uniformly distributed from 12.5% to 66.67% to represent a wide spectrum of options. We chose eight SLAs to analyse how SLA selection changes from the user perspective. We later show that this number is in the 95% confidence interval for being the optimal number of SLAs based on our simulation. We analyse a wider range of 1–60 offered SLAs and how they impact customer conversion from the cloud provider's perspective in Section 6.4. The lines in the background illustrate value progression. *Av* decreases only slightly for the migration SLA, yet the energy savings are significant due to dynamic VM consolidation and PM suspension. For peak pauser SLAs, availability and costs decrease linearly, from high to low values. The *en_savings* values are at first lower than those attainable with the migration scheduler, as the peak pauser scheduler cannot migrate VMs to a fewer number of PMs, but can only pause them for a certain time. With lower availability requirements, however, the peak pauser can achieve higher *en_savings* and lower prices, which could not be reached by VM migrations alone.



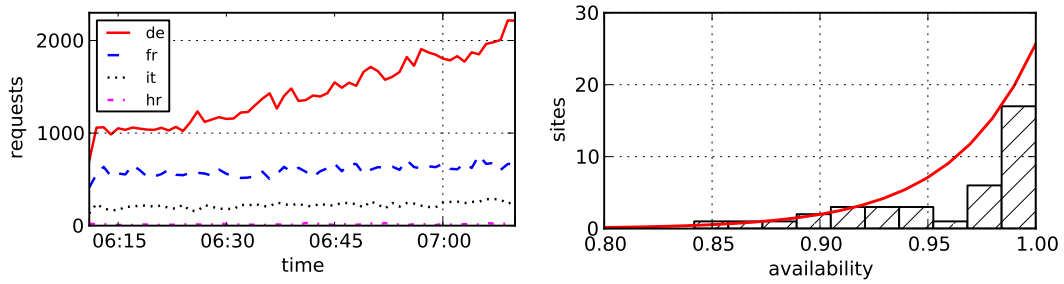
type	m3.xlarge
Av	99.95%
cost	0.28 $/h
service	0.1 $/h

Table 6.2: Base VM

Figure 6.6: SLAs generated for different TCs

6.3 User Modelling

Knowing the SLA offering, the next step is to model user SLA selection in order to analyse the benefits of our progressive SLA specification. We first describe how we derive user requirements and then the utility model used to simulate user SLA selection based on their requirements.



(a) Wikipedia requests per minute

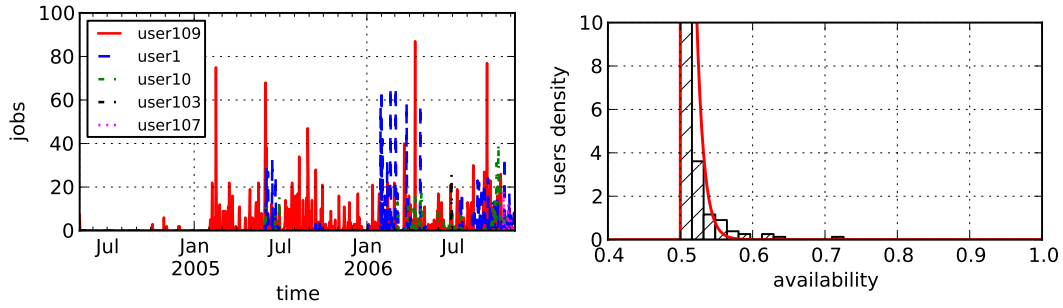
(b) Wikipedia availability distribution

Figure 6.7: Web user modelling

6.3.1 User Requirements Model

To model user requirements, we use real traces of web and high-performance computing (HPC) workload, since I/O-bound web and CPU-bound HPC applications represent two major usage patterns of cloud computing. As we do not have data on availability requirements of website owners, we generate this dataset based on the frequency of end user HTTP requests directed at different websites and counting missed requests, similarly to how reliability is determined from the mean time between failures [146]. A public dataset of HTTP requests made to Wikipedia [173] is used. To obtain data for different websites, we consider Wikipedia in each language as an individual website, because of its unique group of end users (different in number and usage pattern). In this scenario we consider a website owner to be the user of an IaaS service (not to be confused with the end user, a website visitor). The number of HTTP requests for a small subset of four websites (German, French, Italian and Croatian Wikipedia denoted by their two-letter country codes) is visualised in Fig. 6.7 (a) for illustration purposes (we use the whole dataset with 38 websites for actual requirements modelling). The data exemplifies significant differences in amplitudes. Users of the German Wikipedia send between 1k and 2k requests per minute, while the Italian and Croatian Wikipedia have less than 300 requests per minute. Due to this variability, we assume that different Wikipedia websites represent diverse requirements of website owners. We model availability requirements by applying a heuristic – a website's required availability is the minimum necessary to keep the number of missed requests below a constant threshold (we assume 100 requests per hour). Using this heuristic, we built an availability requirements dataset for the web user type from 5.6 million requests divided among 38 Wikipedia language subdomains. The resulting availability requirement histogram can be seen in Fig. 6.7 (b). It follows an exponential distribution (marked in red). There is a high concentration of sites that need almost full availability, with a long tail of sites that need less (0.85–1.0).

For HPC workload, we use a dataset of job submissions made to Grid'5000 (G5k) [15], a distributed job submission platform spread across 9 locations in France. The number of jobs submitted by a small subset of users is visualised in Fig. 6.8 (a). While some users submit jobs over a wide period (*user109*), others only submit jobs in small bursts (*user1, user107*), but the load is not nearly as constant as the web requests from the Wikipedia trace. To model HPC users' availability requirements, where jobs have variable duration as well as rate (unlike web



(a) G5k example job submissions

(b) G5k availability distribution

Figure 6.8: HPC user modelling

requests, which typically have a very short duration), we use another heuristic. Every user's availability requirement is mapped between a constant minimum availability (we assume 0.5) and full availability using $mean\_duration \cdot mean\_rate$, which stands for mean job duration and mean job submission rate per user. Using this heuristic, we built a dataset of availability requirements for the HPC user type from jobs submitted over 2.5 years by 481 G5k users. The resulting availability requirement distribution (normalised such that the area is 1) can be seen in Fig. 6.8 (b). The distribution marked red again follows an exponential distribution (the first bin, cut off due to the zoom level, shows a density of 100), but with the tail facing the opposite direction than the web requirements. HPC users submit smaller and less frequent jobs most of the time, with a long tail of longer and/or more frequent jobs (from 0.5 to 0.75).

Every user's willingness to pay (WTP) is derived by multiplying his/her availability requirement with the base VM price and adding Gaussian noise $\mathcal{N}(0, 0.05^2)$ to express subjective value perception. We selected the noise standard deviation to get positive WTP values considering the availability model. The resulting WTP histogram is shown in Fig. 6.9. It can be seen that HPC users have lower WTP values, but there is also an overlap area with web users who have similar requirements.

6.3.2 Utility Model

The utility-based model is used to simulate how users select services based on their requirements. We use a quasi-linear utility function adopted from multi-attribute auction theory [104] to quantify the user's preference for a provided SLA. The utility is calculated by multiplying the user's SLA satisfaction score with WTP and subtracting the VM cost charged by the provider. The utility for user i from selecting a VM instance type t with availability Av_t is calculated as:

$$U_i(VM_t) = WTP_i \cdot f_i(Av_t) - cost(VM_t) \tag{6.5}$$

where $f_i(Av_t)$ is the user's satisfaction with the offered VM's availability. We model it using a mapping function $f_i : [0,1] \rightarrow [0,1]$, extended from [104] with variable slopes:

$$f_i(Av_t) = \frac{\gamma}{\gamma + \beta e^{Av^2_{offset_i}\alpha(Av_{offset_i} - Av_t)}} \tag{6.6}$$

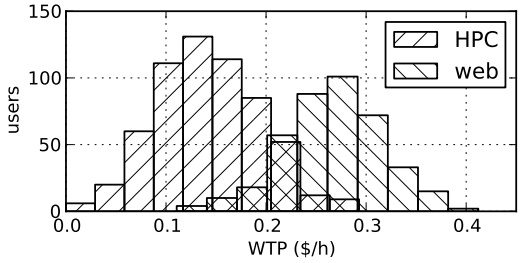


Figure 6.9: WTP histogram

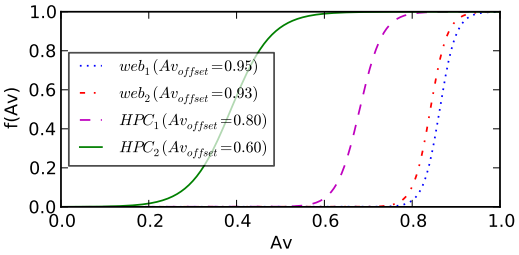


Figure 6.10: SLA satisfaction function

where Av_{offset_i} is the required availability specific to each user, which we select based on the exponential models of Wikipedia and G5k data presented earlier. α, β and γ are positive constants common for all users, which we set to 60, 0.01 and 0.99 respectively. These values were chosen for a satisfaction of close to 1 for the desired availability value and a steep descent towards 0 for lower values, similar to earlier applications of this satisfaction function [104]. The slope of the function also depends on Av_{offset_i}, to model that users who require a lower availability have a wider range of acceptable values. The mapping function is visualised in Fig. 6.10 for a small sample of two HPC and two web users. We can see that for the two web users, the slope is almost the same and very steep (at 0.93 their satisfaction is close to 1 and at 0.8 it is almost 0). The HPC_1 user is similar to the web users, only with lower availability requirements and a slightly wider slope. The HPC_2 user has low requirements and a wide slope from an availability of 0.2 to 0.6. Later in our evaluation, we generate 1000 users, each with different mapping functions, distributed according to the user requirements model.

User i chooses a VM instance type $VM_{selected}$ offering the best utility value:

$$U_i(VM_{selected}) = \max_{\forall t} U_i(VM_t) \tag{6.7}$$

unless all types result in a negative utility, in which case the user selects none. Additionally, we model search difficulty by defining P_{stop}, a probability that a user will give up the search after an SLA has been examined. We model this probability as increasing after every new SLA check, by having $P_{stop_j} = j \cdot check\_cost$, where j is the number of checks already performed and $check\_cost$ is a constant parameter standing for the probability of stopping after the first check. Based on every user's requirements and the SLA offering, min_checks is the minimum number of checks necessary to reach a VM type that yields a positive $U_i(VM)$. We define P_{quit} to be the total probability that a user will quit the search before reaching a positive-utility SLA. By applying the chain probability rule, we can calculate P_{quit} as:

$$P_{quit} = \sum_{j=1}^{min\_checks-1} P_{stop_j} \prod_{k=1}^{j-1} (1 - P_{stop_k}) \tag{6.8}$$

The outer sum is the joint distribution of all possible stop events that may occur for a user and the inner product stands for all event outcomes when searching continued until the j-th event was realised as stopping. We use this expression in the evaluation as a measure of difficulty for users to find a matching SLA.

Parameter	users	web : HPC	#SLAs	runs	*check_cost*	α	β	γ
Value	1000	1 : 1.5	1–60	100	0.015	60	0.01	0.99

Table 6.3: Simulation settings

6.4 Evaluation

In this section we describe the simulation of the proposed progressive SLA specification using user models based on real data traces and analyse the results.

6.4.1 Simulation Environment

The simulation parameters are summarised in Table 6.3. The first step of the simulation is to generate a population of web and HPC users based on the requirement models derived from the Wikipedia and Grid'5000 datasets, respectively. We simulated 1000 users to represent a population with enough variety to explore different WTP and availability requirements. We assume the ratio between web and HPC users of $1 : 1.5$, based on an anlysis of a real system performed in [123]. We determine each user's WTP from the desired availability with Gaussian noise, as already explained in the previous section. The SLA offering was derived from the migration and peak pauser scheduler using the probabilistic modelling technique (Section 6.2). For the examination of SLA selection from the user's perspective, the eight SLAs we already defined in Section 6.2.4 were used. To examine the cloud provider's perspective, we evaluated 1–60 SLAs, doing 100 simulation runs per offering to calculate the most likely optimal number of SLAs. A *check_cost* of 0.015 is selected to initially start with a low chance of the user quitting and then subsequently increase it for every SLA check per Eq. 6.8. The same α, β, γ values that we already explained in the previous section were set that result in a utility of 1 for the required availability and a gradual decline towards a utility of 0 for lower availabilities. The core of the simulation is to determine each user's SLA selection (if any) based on the utility model (Section 6.3).

6.4.2 User Benefits

Simulation results showing the distribution of users among the eight offered SLAs from Section 6.2.4 are presented in Fig. 6.11. Different colours are used for web and HPC users types. It can be seen that most of the users successfully found a service that matches their requirements, with less than 5% of unmatched requests. The majority of HPC users are distributed between SLA 7 and 8 offering 42% and 33% availability, respectively. The majority of web users selected SLA 2, the migration scheduler TC, due to its high availability comparable to a full availability service, but a more affordable price due to the energy cost savings. 26% of web users opted for SLA 3, the peak pauser instance which still offers a high availability (87.5%), but at almost half the price of SLA 2.

The distribution of unmatched users who did not select any of the offered services (where utility was negative for all SLAs) is shown alongside the matched users in Fig. 6.12, showing

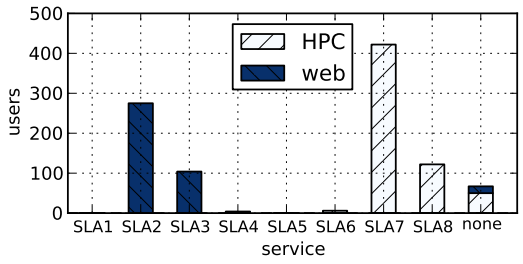


Figure 6.11: Simulated service selection

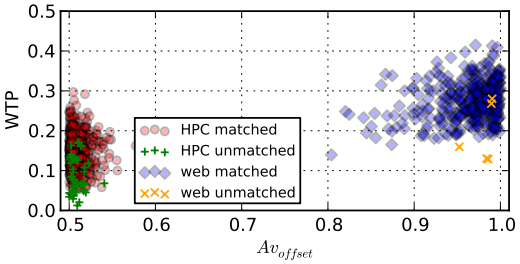


Figure 6.12: Matched and unmatched users

their Av_{offset} and WTP values. We can see that unmatched users have low WTP values, the cause of them not being able to find a suitable service option.

6.4.3 Cloud Provider Benefits

Customer conversion means the number of users who looked at the SLA offering and found an SLA that matches their needs. This metric is an indicator of the provider's economic success. To compare the multiple treatment category system with the traditional way of only having a full availability option, we simulated different SLA offerings. Fig. 6.13 shows customer conversion with colour indicating the selection distribution for different offering combinations of the eight previously examined SLAs. Customer conversion growth can be seen with more service types, due to users having a higher chance of finding a category that matches their requirements. With SLAs 1–2 offered, only SLA 2 was selected, as it still offers a high-enough availability to satisfy user requirements and the price is lower than in SLA 1. As we widen the offering, more SLAs get selected, but the majority of users choose among two SLAs that best suit the two user types that we modelled. Still, a small number of users select other SLAs (SLA 3 and, if offered, SLA 8) which better suit their needs. SLA 5 is never selected due to user requirements and in real clouds such SLAs should be removed to simplify selection.

The introduced service types can be managed in a more energy efficient manner. The average energy savings weighted based on the lease time per VM for the SLA 1-8 offering, compared to the current 99.95% availability Amazon instances represented by SLA 1, are 39%. Full annual lease time was assumed for web users (as web applications are typically running all the time) and was varied based on job runtime and frequency for HPC users (we assume that a VM is provisioned just to perform the submitted job). This shows that more energy efficient management is possible if users declare the QoS levels they require through SLA selection. For the SLA 1-8 scenario, where 5.89× more users can be converted and the annual lease times explained above, a 43% revenue increase is calculated from the service component of the selected VMs. Exact numbers depend on the user type ratio that will vary between cloud providers.

To find the optimal number of offered SLAs, we performed a simulation where we explore customer conversion for a higher number of SLAs. The extra SLAs were generated for the peak pauser scheduler, which allows for arbitrary control of VM availability and price. The peak pauser SLAs were uniformly interpolated between full and no availability to avoid duplicates. Fig. 6.14

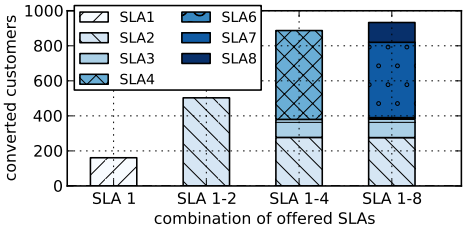


Figure 6.13: Matched users per SLA combination based on WTP.

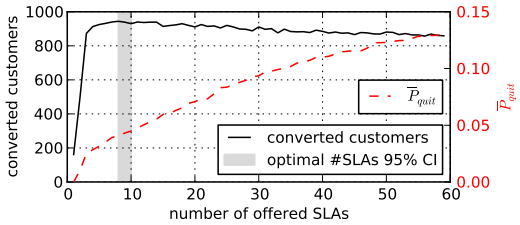


Figure 6.14: Users and $\overline{P}_{quit}$ per SLA number

shows how the number of offered SLAs affects the user conversion count and $\overline{P}_{quit}$, the mean P_{quit} value over all the users (including unmatched ones). After an initial linear growth, we can see that the number of users begins to stagnate and slowly decrease. Once a sufficient offering to satisfy the majority of users is achieved, adding extra SLA options only increases search difficulty. This is seen from the steadily increasing $\overline{P}_{quit}$, the probability that a user will quit the search before finding a positive-utility SLA. For our scenario, the optimal number of converted customers is achieved between 6 and 14 SLAs, depending on the P_{quit} random variable realisations. By applying the bootstrap confidence interval method, we calculate the 95% confidence interval (CI) for the optimal number of SLAs to be between 8 and 10.

6.5 Summary

We presented a novel progressive SLA specification suitable for energy-aware cloud management. We obtained cloud management traces from two schedulers optimised for real-time electricity prices and temperature-dependent cooling efficiency. The SLAs are derived using a method for a posteriori probabilistic modelling of cloud management data to estimate upper bounds for VM availability, energy savings and the resulting VM prices. The SLA specification is evaluated in a utility-based user SLA selection simulation using realistic workload traces from Wikipedia and Grid'5000. Results show mean energy savings per VM of up to 39% due to applying more aggressive energy preservation actions on users with lower QoS requirements. Furthermore, a wider spectrum of user types with requirements not matched by the traditional high availability VMs can be reached, increasing customer conversion.

CHAPTER 7

Performance-Based Pricing in Multi-Core Geo-Distributed Cloud Computing

New pricing policies are emerging where cloud providers charge resource provisioning based on the allocated CPU frequencies. As a result, resources are offered to users as combinations of different performance levels and prices which can be configured at runtime. With such new pricing schemes and the increasing energy costs in data centres, balancing energy savings with performance and revenue losses is a challenging problem for cloud providers. CPU frequency scaling can be used to reduce power dissipation, but also impacts VM performance and therefore revenue. In this chapter, we firstly propose a non-linear power model that estimates power dissipation of a multi-core PM and secondly a pricing model that adjusts the pricing based on the VM's CPU-boundedness characteristics. Finally, we present a cloud controller that uses these models to allocate VMs and scale CPU frequencies of the PMs to achieve energy cost savings that exceed service revenue losses. We evaluate the proposed approach using simulations with realistic VM workloads, electricity price and temperature traces and estimate energy savings of up to 14.57%. This approach is a less invasive approach for reducing power consumption than VM pausing presented in Chapter 4. Frequency scaling is orthogonal to the VM migration controller from Chapter 5, meaning that both approaches could be applied – at the PM and VM level, respectively.

We structure the chapter by first introducing in more details the challenges of frequency scaling in multi-core computers and the inefficiencies of existing control approaches in Section 7.1. This serves as a motivation for our detailed power model for multi-core Intel and ARM CPU architectures in Section 7.2. We then highlight the economical aspects of frequency scaling in cloud computing by explaining emerging VM pricing schemes and propose our own perceived-performance pricing scheme in Section 7.3. We combine the power and pricing models to devise a cloud controller that geographically distributes VMs and applies frequency scaling on PMs in

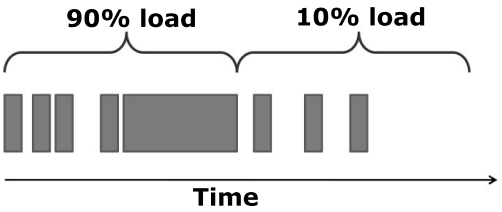


Figure 7.1: Load calculated as a ratio between active and idle CPU for a defined window – dark squares indicate an active CPU.

Section 7.4. In Section 7.5, we present the evaluation methodology and comment on the most significant obtained results.

7.1 Challenges of Multi-Core Frequency Scaling

In this section we introduce the main challenges inherent to multi-core frequency scaling of PMs with multiple CPU cores and motivate our power model and subsequently frequency control approach. We show that neither operating system CPU governors, nor traditional race-to-idle approaches provide optimal energy efficiency because of inaccurate decision making.

7.1.1 Limitations of Current Frequency Governors

The currently used power management system in Linux operating systems is handled by the *frequency governors*, which alter the CPU clock frequency based on a predefined policy. Even though the intention is to reduce the clock frequency when performance is not needed, the approach suffers from limitations.

The metric used to determine the clock frequency in the governors is the system *workload*. The workload is expressed as a ratio between an active CPU and an idle CPU over a given time window, which is illustrated in Fig. 7.1 as two time windows: one with 90% load and one with 10% load.

Workload, however, does not represent the performance, or the "real work" done by an application, but mainly the activity level of the CPU. This means that as long as the CPU is loaded, the performance requirement is recognised as insufficient and the clock frequency is increased to the maximum even though the *actual* performance is sufficient on a moderate clock frequency.

7.1.2 Energy Inefficient Execution

Using workload as the metric for power management decisions often results in race-to-idle scenarios [162, 164], in which the workload is executed as fast as possible in order to obtain an idle system. This execution principle was considered an energy efficient method of executing workload in previous generation single-core microprocessors, because the minimisation of the execution time caused minimal energy consumption.

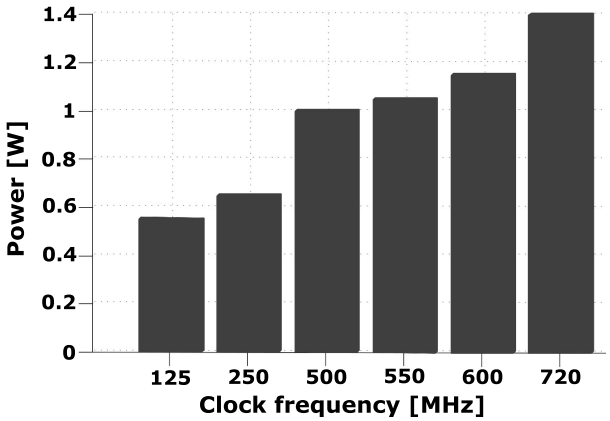


Figure 7.2: Power dissipation for a Cortex-A8 CPU using different clock frequencies.

This is demonstrated in Fig. 7.2, which shows the total power dissipation for a single-core ARM Cortex-A8 processor using different clock frequencies. As seen in the figure, the highest clock frequency (720 MHz) results in roughly 1.4W of power dissipation. When scaling down the frequency roughly 3x (250 MHz), the power dissipation is only reduced by 2x (0.7W), which means that the total energy consumption may be lower when executing at a higher clock frequency.

However, using more recent microprocessors with higher clock frequencies and multiple cores, the power dissipation has increased exponentially – this has reduced the energy efficiency of the race-to-idle principle because the cost in power is greater than the savings in execution time [92, 91, 164]. Fig. 7.3 shows the relative performance-to-power ratio of four different modern platforms. All of the four platforms show an exponential profile, which means that the power dissipation required to operate on the highest clock frequencies is higher than the relative performance gain of the platform. The race-to-idle principle should therefore not be used for energy efficient execution.

7.1.3 Energy Efficient Execution

In order to not race-to-idle, a *performance driven execution* should be used instead of a workload driven execution [92, 91, 164]. By monitoring the actual performance of an application and adjusting the clock frequency accordingly, the energy efficiency can be improved.

Benchmarks were executed on a quad-core ARM platform with an Exynos 5410 SoC using the default *ondemand* governor and the modified performance driven power manager. By using the ondemand governor, the decoder decodes the frames as quickly as possible since the decoding task increases the workload, and the clock frequency is consequently increased. As the frame buffer is filled, the decoder is idle until the frame buffer is emptied by the video display. By instead decoding at the same frame rate as the video display is using (25 FPS), the clock frequency can be reduced to an intermediate clock frequency for the whole execution while still providing the required video quality.

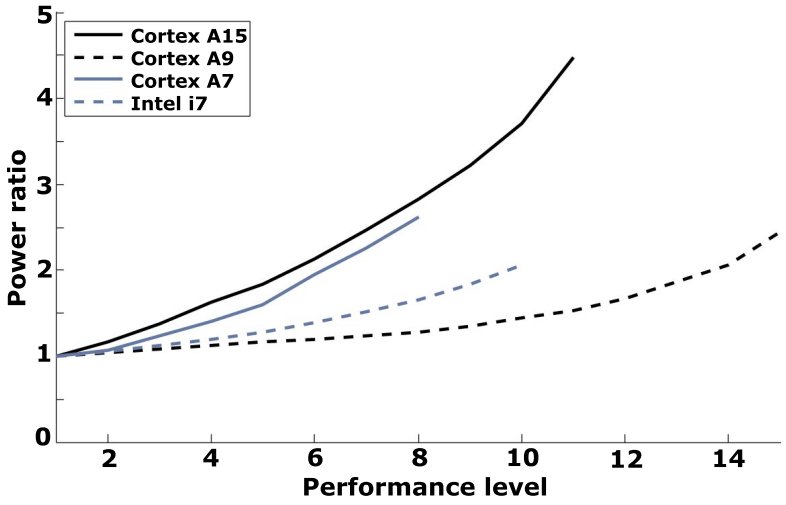


Figure 7.3: Performance to power ratio for different platforms.

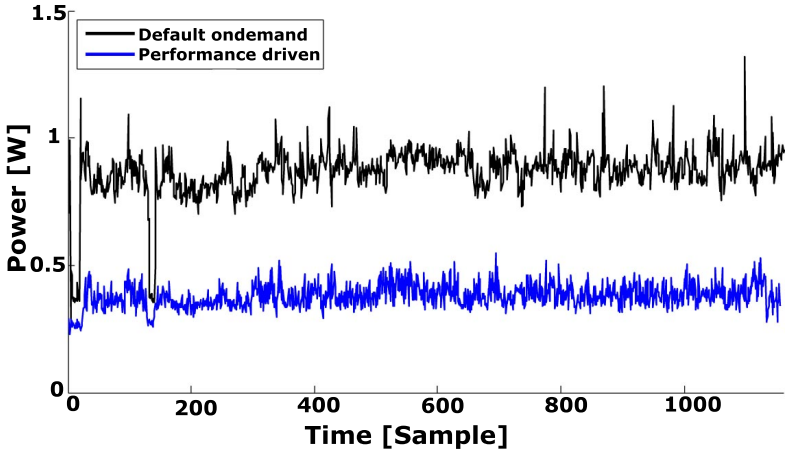


Figure 7.4: Experiments with video decoding using the ondemand governor and performance driven clock frequency scaling on an Exynos 5410 platform [91].

The power dissipation was measured by internal power sensors for both power managers and the result is shown in Fig. 7.4 (more details can be found in [91]). The power dissipation of executing the video decoder using the ondemand governor is shown as the upper, black line and the performance driven power manager is shown as the lower, blue line. Since the standard ondemand governor increases the clock frequency while workload is present, most of the execution demands the highest clock frequencies, which causes excess power dissipation as seen in Fig. 7.4 (upper, black line). By matching the decoding frame rate to the output frame rate (of 25 frames per second (FPS)), the lower clock frequencies are providing enough performance to decode the frames at the intended phase of 25 FPS, and the power is significantly reduced as seen in Fig. 7.4 (lower, blue line).

We intend to bring the performance driven clock frequency scaling using multi-core hardware from CPU level to the cloud level consisting of many parallel machines. The difference is a much more diverse execution platform with additional parameters such as VM migration, network I/O and variable electrical cost models. A power model capable of reflecting such details is needed to get an accurate cost model of the cloud system.

7.2 Multi-Core Power Model

In this section we show how we modelled the behaviour of multi-core CPU power dissipation, accounting for both the CPU frequency and the number of active cores for generic multi-core systems. Such a power model allows us to determine what performance level to execute at, depending on the performance requirements, which is one of the key parts of our cloud controller.

As the power characteristics of modern multi-core CPUs are highly non-linear [92], a non-linear model should be created to accommodate as accurately as possible to the real-world power dissipation. The model should also not be computationally heavy to introduce unnecessary overhead.

7.2.1 ARM and Intel Architectures in the Cloud

Aside from the popular Intel architecture used as a typical server platform, the architecture based on ARM processors made popular through wide usage in smartphones is currently also being investigated for use in servers. ARM processors are much more energy efficient than Intel processors, though their maximum CPU frequency capacity is lower, potentially increasing the necessary number of servers and therefore the communication overhead. The Mont Blanc EU project [157, 82] was devoted to determine whether this approach is valid for large scale cloud platforms. Companies like Calxeda already ship ARM based server machines and Lenovo is pushing its NextScale [28] platform with the motivation to increase the performance-per-watt ratio by focusing on possibly more energy efficient architectures.

We therefore included the ARM architecture in our evaluation as a viable candidate for investigating the effects of performance-based frequency scaling in order to provide a comparison to the Intel architecture.

7.2.2 Power and Energy Consumption Model

For our cloud controller, we created an ARM and an Intel power consumption model. The ARM model was created by reading internal power sensors on an Exynos 5410 board, and the Intel model by using an external measurement device connected directly to the ATX socket on the motherboard. Both models were used in the evaluation, but we describe our modelling procedure with a higher focus on the ARM model for brevity. The same procedure we describe in this section is also applicable to the Intel model (or any other derived model).

We used a similar methodology as the work in [92] to derive the power model, where the model was created by stressing the system to full capacity using all combinations of the clock frequencies and all number of cores. Our power measurements for stressing the ARM board is shown in Fig. 7.5. The figure shows power dissipation of the CPU during full load using different

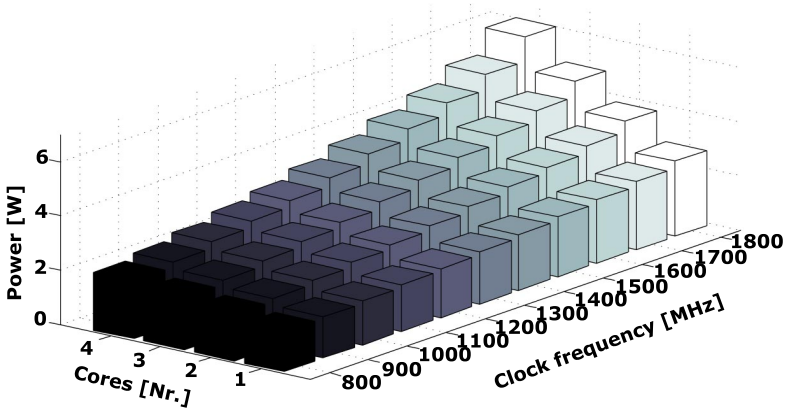


Figure 7.5: A top-down model of a quad-core ARM Cortex-A15 CPU.

configurations of the clock frequency and the number of cores. Naturally, more cores and a higher clock frequency cause a higher power dissipation.

The power measurements were then used as basis for a two dimensional plane fitting algorithm, in order to build a mathematical expression of the multi-core system and its power dissipation. We used a least-squares algorithm [115] provided in Matlab to obtain the polynomial of the form:

$$P_f(q,c) = p_{00} + p_{10}q + p_{01}c + p_{20}q^2 + p_{11}qc + p_{30}q^3 + p_{21}q^2c \tag{7.1}$$

which is a function of the clock frequency (q) and number of cores (c) used. Fig. 7.6 shows the analytical representation of the power dissipation and the data points obtained from Fig. 7.5 for the ARM platform. The clock frequency and the number of cores used are represented as discrete steps from 1 to 11 and from 1 to 4 respectively. The plane shown in Fig. 7.6 was fitted to the data values using the obtained parameters shown in Table 7.1. The same method was used for the Intel platform, and other parameters were then obtained.

By comparing the model to the measurements, a maximum deviation of 18.7% was obtained and an average deviation of 6.4% compared to the experimental data, which we considered feasible for our cloud controller evaluation.

The idle power was modelled similarly to Eq. 7.1, but without the core component c as:

$$P_{idle} = p_{00} + p_{10}q + p_{20}q^2 + p_{30}q^3 \tag{7.2}$$

where q is the clock frequency step and the p-parameters are identical to the fitted parameters in Table 7.1.

p_{00}	p_{01}	p_{10}	p_{11}	p_{20}	p_{21}	p_{30}
1.318	0.03559	0.2243	-0.00318	0.03137	0.000438	0.00711

Table 7.1: Power model coefficients.

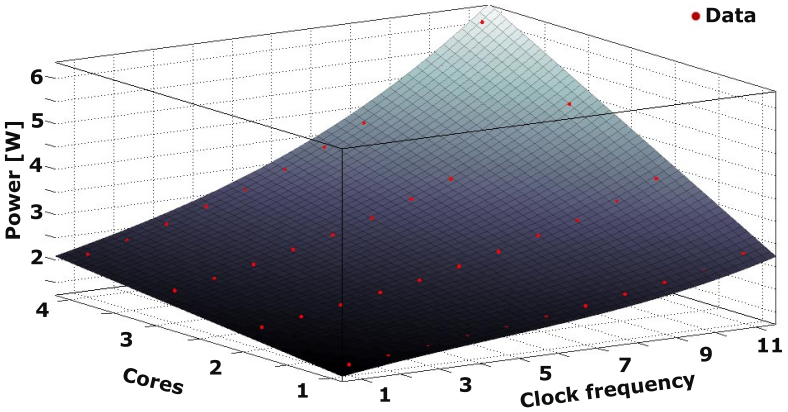


Figure 7.6: A mathematical representation of the power values in Fig. 7.5 obtained using surface fitting methods.

7.2.3 I/O-based Power Dissipation

The power dissipation of a PM varies depending on the CPU utilisation of the machine, which is dependent of the I/O usage of the workload. A *CPU-boundedness parameter (β)* is therefore used to model the portion of the execution which consists of low intensity I/O operations. The parameter may range between 0 and 1 to represent workloads of different CPU-boundedness properties with values close to 0 corresponding to I/O-intensive workloads and values close to 1 corresponding to CPU-bound workloads [76]. The value of β normalises the ratio between low intensity I/O bound and high intensity CPU bound instructions in the workload.

The power model was therefore extended to account for the CPU utilisation based on the VM CPU-boundedness of the executing workload. To do so, the CPU utilisation u is expressed as:

$$u = \sum_{core} \frac{\gamma_{core}}{cores_{active}} \tag{7.3}$$

where γ_{core} is a power ratio depending on the VM CPU-boundedness β and $cores_{active}$ represents the number of the currently used cores of the PM.

Similarly to the basic power model (Eq. 7.1), the power dissipation was evaluated on the same platform with different β in order to train the model. The experiments were run using the *Berserk* benchmark on a single active core (to avoid any lack of scalability from using multiple cores and get the pure effect of only the I/O). The Berserk benchmark framework we also used in Chapter 4 was extended [16] for stressing the CPU cores at various CPU-boundedness ratios β. The workload itself is again a CPU-intensive task – repeated recursive calculation of Fibonacci numbers, only parallelised to execute on all available CPU cores. By passing different β parameters to the benchmark, proportional ratios of the workload are deferred to a remote server, making the work less or more CPU-bound for monitoring purposes. For example for a value of β equal to 0, all the work is sent to and received from a remote server via the network, making the task fully I/O-bound. For a value of β equal to 1, all the work is executed locally, resulting in a CPU-bound task.

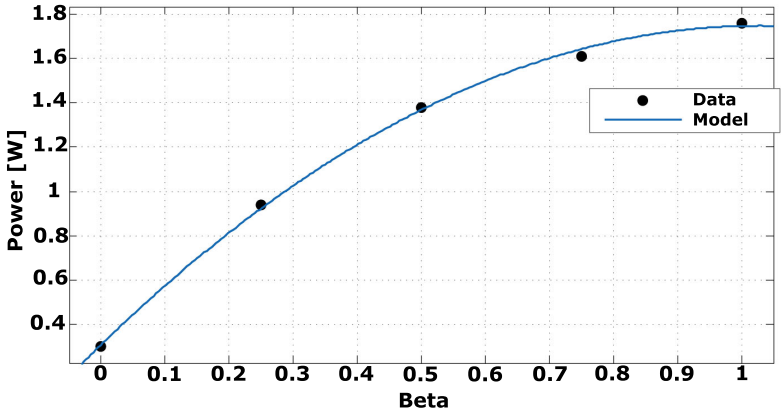


Figure 7.7: Measurements of power dissipation based on I/O expressed as a ratio β. The model is expressed as a second degree polynomial.

The explored I/O ratios (β values) were selected in the range [0.0, 0.25, 0.5, 0.75, 1.0] where 0.0 indicates total I/O blocking and 1.0 indicates no I/O (a fully CPU-intensive workload). The CPU (in this case the ARM architecture) was executing at 1600 MHz for all measurements (we show the behaviour of the model at different CPU frequencies in the following experiment), and the measurement results are shown in Fig. 7.7 as the data values.

A one-dimensional curve fitting technique was used to model the power ratio γ_{core} as a second degree polynomial:

$$\gamma_{core} = \frac{p_o\beta^2 + p_1\beta + p_2}{P_{max}} \tag{7.4}$$

where β is the CPU-boundedness of the core, P_{max} is the maximum power dissipation of a core and the obtained function parameters are listed in Table 7.2. The curve in Fig. 7.7 shows the model of the γ_{core} function.

The accuracy of the γ_{cores} function at different CPU frequencies was evaluated in another experiment. We executed the Berserk benchmark with the same β parameters at clock frequencies 1600MHz, 1200MHz and 800MHz. Both the measurement data and the model for each experiment is shown in Fig. 7.8, in which the circles are measurement points. The maximum difference between the data and the model was 10.59% and the average difference was 4.32%, which we considered as acceptable for our cloud controller evaluation.

p_0	p_1	p_2
-1.362	2.798	1.31

Table 7.2: Coefficients for the power ratio γ_{core} model.

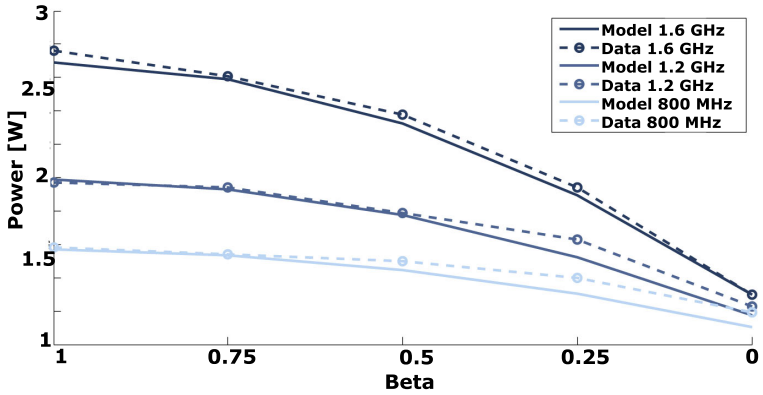


Figure 7.8: Verification of the I/O-based power model from Fig. 7.7. Verification performed with three different frequency levels.

To combine all of the presented model components, the power dissipation P of a PM in the cloud system was modelled as:

$$P = P_{idle} + (P_f - P_{idle})u \tag{7.5}$$

where P_{idle} is the idle power expressed in Eq. 7.2, P_f is the active power expressed in Eq. 7.1 and u is the utilisation modelled in Eq. 7.3 based on the I/O activity parameter β. The model to compute the power dissipation of a PM in the cloud system was based on the approach proposed in [123], where PM's power dissipation is linearly related with its CPU utilisation. The model was extended to take into account the active power dissipation of the currently used cores of the PM at the operating frequency and the CPU-utilisation of the cores based on the VM CPU-boundedness β.

7.2.4 Energy Calculation

With the multi-core, frequency-dependent power model, we now present the details of electricity cost calculation based on geotemporal inputs. This conversion of power dissipation to a monetary value is crucial for comparing potential energy savings with revenue losses under performance-based VM pricing that we explore in the next section. This part of the model was described in more detail in Section 5.2.2, so we just summarise the important expressions necessary to understand the integration with other model components from this chapter.

The power model so far was expressed for instantaneous values, but to express the energy costs for the cloud provider we need to add a time dimension to account for geotemporal inputs and actions like frequency scaling. Therefore, we time-stamp the expressions that change as time progresses, so for example P_t is the power dissipation for a PM at time t (as it depends on the current CPU frequency f). We define an observed time period of N equidistant time stamps in the range from t_0 to t_N, denoted $[t_0, t_N]$.

Cooling overhead based on local temperatures is derived from the power signal of each PM at its corresponding data center location. For this purpose, the model for computer room

air conditioning using outside air economisers from [180] was applied. Cooling efficiency is expressed as partial PUE – $pPUE_{dc,t}$ at data center dc at time t, which affects the power overhead based on the following formula:

$$P_{tot,t} = P_t + P_{cool,t} = pPUE_{dc,t} \cdot P_t(pm) \tag{7.6}$$

where $P_{cool,t}$ is the power necessary to cool the physical machine, and $P_{tot,t}$ stands for the combined cooling and computation power. For each data center location dc, there is a time series of temperature values $\{T_{dc,t} : t \in [t_0, t_N]\}$. The dynamic value of $pPUE_{dc,t}$ is modelled as a function of temperature T to match hardware specifics as:

$$pPUE_{dc,t} = 7.1705 \cdot 10^{-5} T_{dc,t}^2 + 0.0041 T_{dc,t} + 1.0743 \tag{7.7}$$

These power signals are then integrated over time and combined with fixed or real-time electricity prices (both models are explored in the evaluation) for the corresponding data center location to compute the total energy cost C_{en} of every individual PM.

$$C_{en} = \frac{t_N - t_0}{N} \sum_{t=t_0}^{t_N - 1} P_{tot,t} e_{dc,t} \tag{7.8}$$

The integration is approximated using the rectangle numerical integration method. At each dc location, there is a time series of electricity prices $\{e_{dc,t} : t \in [t_0, t_N]\}$.

7.3 Performance-based VM Pricing

After covering the detailed components that make up energy costs in modern multi-core physical machines, in this section we continue analysing the economical side of cloud computing by looking at VM pricing. Concretely, we cover state of the art performance-based VM pricing schemes used by providers such as ElasticHosts and CloudSigma where the user pays for the VM proportionally to the allocated CPU frequency. We then show on a practical experiment that these schemes do not account for properties like the CPU-boundedness of the VM's workload and its effect on QoS in the price calculation. To address such drawbacks, we present our own perceived-performance VM pricing scheme as a next step in performance-based pricing, adapted for both Intel and ARM architectures. Having models for both the energy costs and VM revenue accounting for frequency scaling on multi-core PMs will allow us to explain our cloud controller in the next section.

7.3.1 Emerging Performance-based Pricing Cloud Providers

In the performance-based pricing model used by several cloud providers, each user is charged on a per-time-unit basis (e.g. hourly) depending on the provisioned resources and their characteristics. The overall cost includes the cost for CPU provisioning, the allocated amount of RAM and the use of other resources, e.g. storage. Such a pricing scheme is offered by several providers, such as ElasticHosts [18] and CloudSigma [17], that allow the provisioning of different CPU frequency and core quantities, calculating the total CPU capacity allocated for the final invoice. This enables

users to choose between equivalent combinations of frequencies and number of virtual CPU cores that incur same CPU provisioning costs [18].

Based on the performance-based pricing scheme offered by ElasticHosts and CloudSigma, we fitted a pricing model that describes the behaviour of both schemes, similarly to the work in [150] where the ElasticHosts pricing scheme was initially modelled and analysed. In our obtained model, the price charged for CPU provisioning changes linearly with the total requested CPU capacity, as CPU capacity can be customised for different selected CPU frequencies and number of cores. Also, we extended the model to describe the RAM allocation. As VMs may have different RAM capacity requirements, the price varies according to the selected RAM size. Finally, the cost for other resources used is considered to be fixed in the model as it is not the focus in this work. Hence, the price C_{vm} of each VM at frequency f_{CPU} is computed as:

$$C_{vm} = C_{base} + C_{CPU} \sum_{cpu \in vm} \left(\frac{f_{cpu} - f_{base}}{f_{base}}\right) + C_{RAM} \frac{RAMsize}{RAMsize_{base}}, \tag{7.9}$$

where C_{base} is the price of the VM at minimum capacity, i.e. a CPU at minimum frequency f_{base}. C_{CPU} and C_{RAM} are cost weights used to generate the VM price for different CPU and RAM capacities. By replacing these variables with actual constants (presented later in the evaluation), the prices for configurations offered by ElasticHosts or CloudSigma can be approximated.

7.3.2 Workload Heterogeneity Implications

While the performance-based pricing model offered by ElasticHosts and CloudSigma enables the cloud provider to balance energy savings with the revenue losses from actions such as CPU frequency scaling, it ignores the impact of VM workload characteristics. We illustrate this in an empirical experiment we have performed to show how operating CPU frequency may affect workload performance in a different way depending on the application's CPU boundedness (β) characteristics.

We executed the Berserk benchmark (already explained in Section 7.2.3) on one local server with a *remote_ratio* parameter indicating the portion of the work to offload to a different, remote server. The rest of the tasks were executed locally. Both servers had the same Quad-core 2.6 GHz AMD Opteron 4130 processor. The *remote_ratio* parameter therefore controlled the workload's CPU boundedness, as we could control if the task was more CPU-bound (i.e. a low *remote_ratio*) or more I/O-bound where they would wait on the results to arrive from a network resource (i.e. a high *remote_ratio*). We calculated the CPU boundedness parameter β as inversely proportional to *remote_ratio*. This approach enabled us to set arbitrary workload CPU boundedness. The experiment was run for six equidistant β values between 0 and 1.

To also measure the effects CPU frequency scaling has in this context, we executed each of the workload characteristics on all five CPU frequency levels (applied both locally and to the remote server) that our server offered using the *cpufrequtils* tool (the scripts we developed are included together with the Berserk benchmark [16]). We collected the duration of the benchmark under each of the workload CPU boundedness β and server CPU frequency combination. The

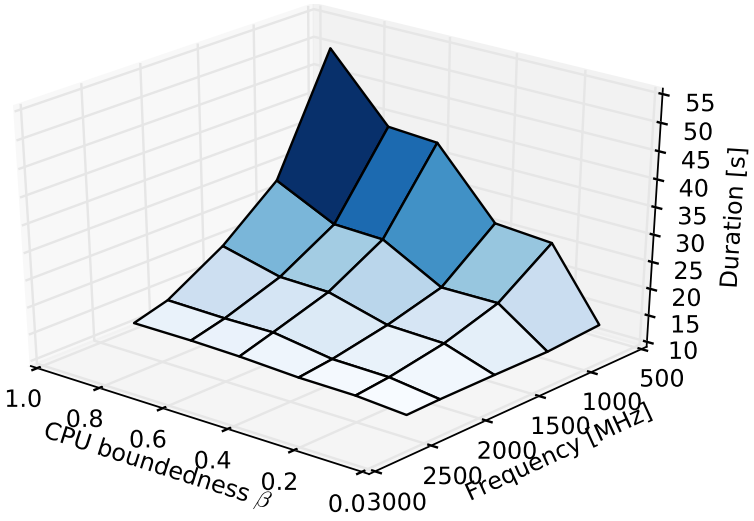


Figure 7.9: Benchmark duration for different workload CPU-boundedness (β) parameters and server CPU frequencies.

energy consumption was collected through an Eaton Wattmeter[1] using its SNMP API (the energy measurement source code is part of our Philharmonic simulator [72]).

The results are shown in Fig. 7.9. As can be seen, the execution time of an application with high CPU-boundedness (β) increases significantly when using a lower frequency and remains mostly unaffected for application with a lower CPU-boundedness (β close to 0). Using the current performance-based pricing for CPU provisioning, like ElasticHosts and CloudSigma, a low frequency for jobs with low CPU boundedness would result in significantly lower revenue for the provider, even though the application performance would not be greatly affected. This was the main motivation for our perceived-performance pricing scheme that we present in the following section.

7.3.3 Perceived-Performance Pricing

As mentioned earlier, performance-based pricing, used by cloud providers like ElasticHosts and CloudSigma, does not consider the impact of frequency reduction on VM performance. To do so, Eq. 7.9 is extended for pricing based on the performance perceived by the hosted application based on the approach we proposed in [127] by defining f_{cpu} as:

$$f_{cpu} = \beta f + (1-\beta) f_{max}, \tag{7.10}$$

where f_{max} is the maximum operating frequency of the host so that CPU-bound applications incur lower CPU provisioning cost when using lower frequencies that may result in lower performance. On the other hand, I/O-bound applications that are not affected by the change in frequency do not receive significant decrease in cost. It is assumed that the impact of frequency on application performance is same for each core of the VM (β).

[1] EATON ePDU PW104MA0UC34

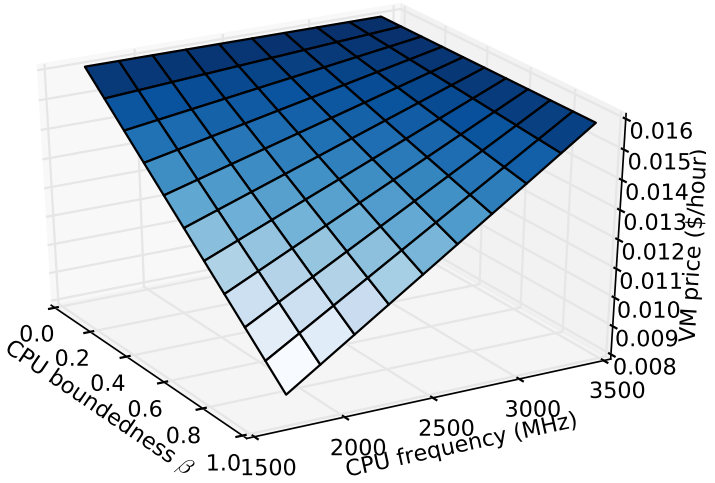


Figure 7.10: Perceived-performance pricing model.

The pricing model is presented in Fig. 7.10, where the axes show the provisioned CPU capacity, the CPU boundedness of the VM (parameter β) and the respective VM price. The CloudSigma model constants were used. The amount of RAM was not varied. CPU frequency values for the Intel platform were used. We assumed four active CPU cores where the CPU frequency is scaled linearly (so the sum of assigned CPU frequencies of all the cores was used in the price calculation).

The linear curve for $\beta = 1$ corresponds to the case of performance-based pricing where the CPU provisioning cost changes linearly with the actual CPU frequency ($f_{cpu} = f$). As the value of β is decreased, the price is less affected by CPU scaling (becoming constant for $\beta = 0$), which matches the workload behaviour from the experiment described earlier.

It is assumed that resources are charged on an hourly basis. The total service revenue is computed by adding the per-hour provisioning cost of Eq. 7.9 for each VM served, allowing us to compare it with the frequency-based energy costs from Section 7.2.

7.3.4 Prices for Different Architectures

As mentioned in Section 7.2.1, besides using Intel architectures with ElasticHosts and CloudSigma pricing, in our work we are also interested in analysing the new pricing models on ARM architectures. Since the performance of an ARM-based CPU is significantly lower than the Intel, the price was scaled according to this factor. The authors in [62] evaluated the ARM Cortex-A15 against a Haswell i5 executing high efficiency video coding (HEVC) decoding tasks and obtained a 6x performance advantage for the Intel. Similarly, the authors in [82] evaluated a Sandy-Bridge based Xeon and a Cortex-A15 using various benchmarks, with roughly an 8x performance advantage for the Intel. Finally, a set of scientific benchmarks was used in [147] with both a Cortex-A15 CPU and a Sandy-Bridge based high-end i7 CPU. The results indicated that the i7 performed roughly 16x better than the ARM.

Based on these numbers, and by using one ARM Cortex-A15 CPU and one Sandy-Bridge based high-end i7, we normalised the performance values to the clock frequency of both platforms in order to match with the cost model of the cloud provider. We assumed a 11x performance advantage for the Intel architecture, and therefore we assumed a 11x cost reduction in the cloud service when using the ARM platform.

7.4 Cloud Controller Description

Having described both the multi-core, geographically-dependent energy consumption model used to compute the energy costs from operating the cloud and the perceived-performance pricing scheme used to compute the revenue from VM provisioning, in this section we explain our cloud controller. The cloud controller balances both of these cost-related components to obtain a quantitative comparison of energy saving and revenue loss trade-offs, which can be addressed as an optimisation problem. In other words, both cost-related components addressed in this chapter are used to determine the actions invoked by the cloud controller.

We describe the BCFFS cloud controller that determines the VM migration and frequency scaling actions to be triggered in order to achieve energy cost savings that exceed the revenue losses. As these two control actions – VM migration and CPU frequency scaling – are mutually orthogonal, they are considered as two complementary actions in order to optimise the allocation and configuration of VMs to PMs. Hence, the two actions are examined in the algorithm separately as two stages, firstly migrating the VMs to PMs and then scaling the CPU frequencies of the PMs to achieve further energy cost savings. During the VM migration stage, the controller migrates VMs to PMs so that resource utilisation is maximised while preferring more economical locations in terms of electricity and cooling costs. Then, the controller reduces the CPU frequencies of the PMs iteratively as long as the energy cost savings exceed the service revenue losses. The algorithm is invoked periodically to trigger appropriate actions. In a real cloud system the algorithm could be automatically invoked by new VM arrivals or threshold violations of geotemporal inputs, e.g. a temperature increase of 1 C. Next, the two stages of the algorithm are described in more detail.

7.4.1 VM Migration Stage

During the first stage, the controller allocates newly requested VMs or reallocates VMs from underutilised hosts using migration based on the power overhead and the geotemporal input parameters of the PMs. As the underlying bin packing problem of VM allocation is NP-hard, for this stage we propose the heuristic polynomial time BCF algorithm we presented as Alg. 3 in Chapter 5.

To briefly summarise the logic of the algorithm once again, first all the newly requested VMs or VMs that run on underutilised hosts (as discussed in Chapter 5) are selected for allocation. The selected VMs are then migrated, prioritising VMs larger in their resource requirements (e.g. more required RAM, CPU cores). The host PM of each VM is selected by preferring active (already hosting another VM), smaller machines (in order to minimise the idle power overhead)

Algorithm 4 Frequency Scaling Stage.

Ensure: Reduce CPU frequencies while energy savings exceed revenue losses.

1: **procedure** FREQUENCY SCALING STAGE
2: $decrease\_feasible \leftarrow False$
3: reset frequency of $\forall pm \in active$ to f_{max}
4: **for** $pm \in active$ **do**
5: $f \leftarrow f_{max}$ ▷ Start the loop at max frequency
6: $revenue\_cur \leftarrow get\_revenue(pm, f_{to\_apply})$ ▷ Service revenue, $\forall vm \in pm$
7: $en\_cost\_cur \leftarrow get\_en\_cost(pm, f_{to\_apply})$ ▷ Energy cost of the pm
8: **while** $f > f_{min}$ **do**
9: $f \leftarrow f - f_{step}$
10: $revenue\_new \leftarrow get\_revenue(pm, f)$ ▷ Revenue for the new frequency
11: $en\_cost\_new \leftarrow get\_en\_cost(pm, f)$ ▷ New energy cost
12: $revenue\_loss \leftarrow revenue\_cur - revenue\_new$
13: $en\_savings \leftarrow en\_cost\_cur - en\_cost\_new$
14: **if** $en\_savings > revenue\_loss$ **then**
15: $revenue\_cur \leftarrow revenue\_new$ ▷ Update current service revenue
16: $en\_cost\_cur \leftarrow en\_cost\_new$ ▷ Update current energy cost
17: $decrease\_feasible \leftarrow True$
18: $f_{to\_apply} \leftarrow f$ ▷ Update currently selected frequency
19: **else**
20: break
21: **end if**
22: **end while**
23: **if** $decrease\_feasible$ **then**
24: apply f_{to_apply} to pm
25: **else**
26: remove from $active$: $\forall \hat{pm} \in PMs$ s.t. $\hat{pm}$ has higher mean β_{vm} and lower electricity price and temperature than pm
27: **end if**
28: **end for**
29: **end procedure**

and data centers with a lower combined electricity price and cooling overhead cost based on the geotemporal input prices model outlined in Section 7.2.4.

7.4.2 Frequency Scaling Stage

Having allocated the VMs to PMs, the CPU frequencies of the PMs are adjusted in the next stage. We assume that each host can operate between a minimum and maximum frequency, f_{min} and f_{max} respectively. The appropriate CPU frequencies are selected based on both the geotemporal inputs and the workload characteristics, by considering their overall impact on the cost components presented in Sections 7.2 and 7.3. To do so, a PM's CPU frequency is reduced

only when energy savings from the reduction in the CPU frequency exceed the revenue losses under perceived-performance pricing. The algorithm is described in Alg. 4. From a high level, the CPU frequency of each PM is initially set to its maximum frequency f_{max} (line 3). Then, the algorithm iterates through the list of $active$ PMs (line 4) to determine the most efficient CPU frequency for each one, analysing the range of the available CPU frequencies (line 9). Note that the actions determined in each step do not have to be executed physically before the procedure halts, after which the final PM frequencies can be determined and applied.

To pick the best CPU frequency, the cost-related components for the current PM are calculated for the previously determined and the next lower frequency (lines 10–11). The components include the service revenue from the VMs allocated to the current PM and the PM's energy cost based on the multi-core power model and energy cost calculation presented in Section 7.2. Whenever the consideration of the lower CPU frequency results in energy cost savings which exceed the subsequent revenue loss, the new frequency is chosen for the current PM (line 14) and the algorithm continues to the next lower available frequency (line 9). The procedure in the inner loop terminates when the revenue losses exceed the respective energy savings (line 20). If no frequency reduction occurred for the PM ($decrease\_feasible$ stays $False$), the procedure will remove PMs with higher average β and lower electricity price and temperature (line 26) before continuing. The idea is that such PMs may incur even lower energy savings and higher revenue losses, hence they can be omitted from the analysis to prune the search space.

7.5 Evaluation

In this section we evaluate the presented cloud controller in a simulation that we first describe as part of the evaluation methodology. We then proceed with presenting the simulation results showing the impact of our cloud controller with a focus on different environment factors.

7.5.1 Methodology

The BCFFS method is evaluated in a simulation of 2k VMs based on real traces of geotemporal inputs and VM CPU-boundedness values. The goal of the evaluation is to show the cost savings attainable using our approach, the impact on service revenue and to analyse the dependence on external factors, such as electricity prices and VM workloads. The simulations were executed on the Philharmonic simulator described in Section 5.4.1. The simulated controller is called to determine cloud control actions, such as VM migrations or PM frequency scaling.

To compute the energy costs of the simulated geographically distributed cloud, we consider the same use case of six data centers as in Section 6.2.1. A dataset of real-time electricity prices described in [33] and temperatures from the Forecast [19] web service were again used (with synthetically generated data for cities where we had no RTEP data). The data center locations used in the simulation (Fig. 6.2) were selected to resemble Google's deployment. Additionally, a scenario with fixed electricity prices over time is considered in the evaluation, using the mean values for each location.

The simulator was set up using the infrastructure parameters shown in Table 7.3. The table shows two architecture types: ARM and Intel. Their respective performance characteristics were

Architecture	PMs	VMs	f_{min}	f_{max}	f_{step}
ARM	2k	2k	0.8 GHz	1.8 GHz	100 MHz
Intel	2k	2k	2.6 GHz	3.4 GHz	200 MHz

Table 7.3: Infrastructure parameters.

Pricing Model	C_{base}	C_{CPU}	C_{RAM}	$RAMsize_{base}$
ElasticHosts	0.027 $/h	0.018 $/h	0.025 $/h	1 GB
CloudSigma	0.0045 $/h	0.0017 $/h	0.004 $/h	1 GB

Table 7.4: Pricing model parameters.

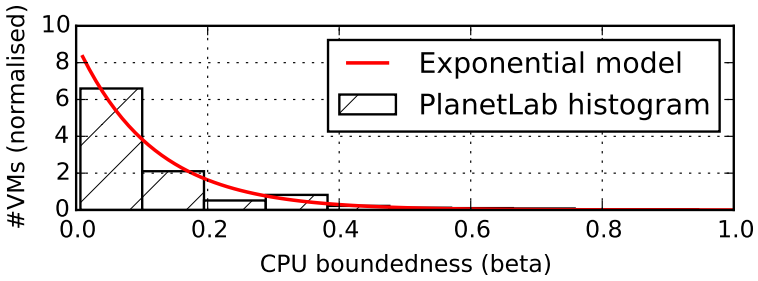


Figure 7.11: VM CPU-boundedness distribution from PlanetLab traces.

derived from the real specifications, such as minimum CPU frequency f_{min}, maximum CPU frequency f_{max} and the absolute frequency increase or decrease step size f_{step}. The parameters we fitted for the pricing models in Section 7.3 to calculate hourly VM prices based on the pricing schemes offered by ElasticHosts [18] and CloudSigma [17] are shown in Table 7.4. The cost of other resources which is not the focus in this work, e.g. disk, was considered to be fixed. We show results for both CPU architectures and both pricing schemes only in sections where we compare the effects of these respective factors on the attainable energy and cost savings. Other presented results are limited to the ARM architecture and the CloudSigma pricing scheme, which proved to be more promising for the application of our method, as will be shown later.

Each run simulated the cloud system for seven days of operation (168 h) with an hourly step size (1 h). The step size was chosen based on the available datasets of geotemporal inputs. However, note that different time intervals and triggering events, e.g. thresholds in geotemporal input changes or new VM arrivals could invoke the cloud controller in production environments. The characteristics of each resource considered in this work, namely the number of CPU cores and the amount of RAM, were uniformly distributed. Heterogeneous VMs were assumed with 1 or 2 CPU cores and RAM capacity ranging between 8 and 16 GB RAM in order to model different VM requests and prices. Each PM consists of 1–4 CPU cores and 16–32 GB RAM to model specification diversity. For each VM, the boot time and duration were varied using a uniform distribution to generate random values within the simulation time and distribute delete events over the simulation period and range the utilisation of the resources.

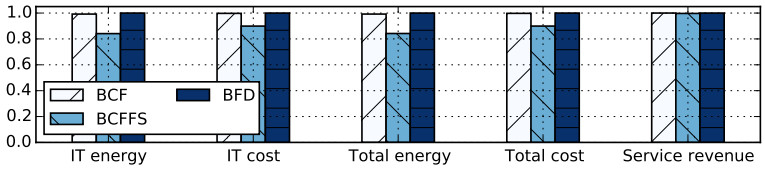


Figure 7.12: Aggregated results for a 2k Intel PM simulation.

The CPU-boundedness of each VM was modelled based on the CPU usage traces from the PlanetLab dataset [12]. The dataset includes CPU usage traces of 1024 VMs. The data was collected every five minutes throughout a day. To generate realistic VM CPU-boundedness values in the simulation, the average CPU usage of each VM in the dataset was calculated and mapped to a β value. From the generated dataset of β values, an exponential distribution was fitted. The distribution is shown in Fig. 7.11. The figure also includes the empirical histogram of the traces normalised to an area of 1. The β values of the VMs used in the simulation were generated based on this model.

We consider two baseline controllers for results comparison. The BFD algorithm developed in [44] (also used as the baseline in Chapter 5) is a cloud controller that migrates VMs, dynamically adapting to user requests. The second baseline controller, BCF, is a variant of the BCFFS controller that only applies the VM migration stage based on geotemporal inputs, but does not consider frequency scaling. The BCFFS controller allows us to quantify the improvement brought by CPU frequency scaling in isolation.

The remainder of the section presents individual results for the different simulation scenarios we performed to compare the energy and cost savings and the performance implications from applying the proposed cloud controller approach. The parameters specified earlier are used in all of the experiments, unless otherwise stated.

7.5.2 Cloud Controller Evaluation

We begin by showing the cloud controller evaluation and comparison to the baselines for the Intel architecture. Fig. 7.12 includes the aggregated results for the achieved energy costs and service revenue – the energy and cost used by the IT equipment, total energy and cost which include the cooling overhead taking into account the outside temperatures and the service revenue from hosting VMs considering the perceived-performance pricing model described in Section 7.3.3. The values are normalised as a relative value of the results obtained using the baseline BFD controller, while the absolute values can be found in Table 7.5. It can be seen that the BCFFS controller achieves 10.06% energy savings compared to the baseline controller BFD, out of which 9.86% are the additional energy savings achieved by using frequency scaling (the savings compared to the BCF controller). The service revenue losses from using perceived-performance pricing were not significant with a drop of less than 0.3%, compared to the BFD baseline. This is because the frequency scaling algorithm presented in the previous section does not scale frequencies if the revenue loss exceeds the energy cost savings.

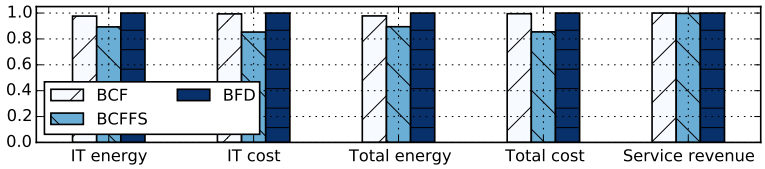


Figure 7.13: Aggregated results for the ARM power and pricing model.

7.5.3 Architecture Impact

Having shown the results for the Intel architecture, we now show results for the same simulation, only this time using the ARM power model, presented in Section 7.2.2. This allows us to compare the architecture impact on attainable energy cost savings. As we previously mentioned, ARM processors are increasingly popular due to their good computation-per-watt ratio and are being explored for use in data centers as part of the Mont Blanc project [82]. The VM pricing is also adapted for the lower ARM performance compared to Intel, as detailed in Section 7.3.4.

The results for the ARM architecture are shown in Fig. 7.13. Even higher savings are achieved than for the Intel architecture – the BCFFS controller achieves 14.57% energy cost savings compared to the BFD baseline and 14.13% compared to the BCF baseline. Even lower service revenue losses (less than 0.25% drop compared to BFD) again indicate that the impact on VM performance is not significant. Given the better applicability of our controller method to ARM architectures, we limit the remainder of the results to this architecture. Absolute values can be found in Table 7.6.

	BCF	BCFFS	BFD
IT energy (kWh)	18793.73	15933.10	18943.00
IT cost ($)	974.60	878.50	977.00
Total energy (kWh)	22501.25	19095.69	22678.65
Total cost ($)	1161.18	1046.75	1163.89
Service revenue ($)	6543.78	6524.54	6543.78

Table 7.5: Absolute aggregated Intel simulation results.

	BCF	BCFFS	BFD
IT energy (kWh)	681.79	623.50	698.19
IT cost ($)	39.23	33.68	39.48
Total energy (kWh)	817.12	747.47	835.61
Total cost ($)	46.78	40.17	47.02
Service revenue ($)	588.81	587.33	588.81

Table 7.6: Absolute aggregated ARM simulation results.

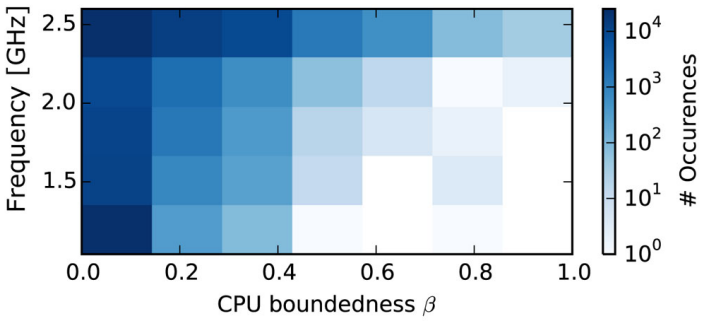


Figure 7.14: Occurrences of (β, f) combinations among the controlled VMs for the ARM architecture.

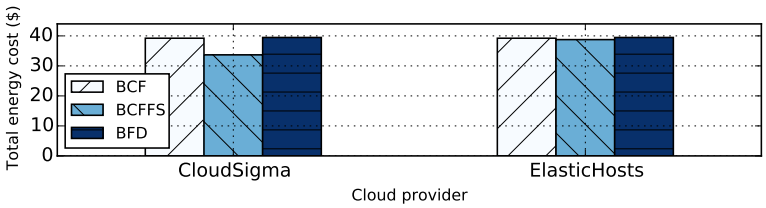


Figure 7.15: Energy costs for the pricing models used by different cloud providers.

7.5.4 Dynamic CPU Frequency Analysis

To explore the frequencies f assigned to VMs dynamically during the simulation and compare them with the VMs' CPU boundedness β, we counted the number of occurrences of each (β, f) combination for every VM and time slot. This data is illustrated as a bivariate histogram in Fig. 7.14 with the number of occurrences shown on a logarithmic scale. Darker areas indicate a higher number of frequency occurrences for the respective (β, f) combination. It can be seen that the occurrences of CPU frequencies assigned based on each VM's CPU boundedness match the areas where VM prices are high, based on the perceived-performance pricing model from Fig. 7.10. The area with high β and low f, where prices would be the lowest, contains no occurrences. The darkest areas of the graph with a high number of occurrences represent the balance between energy savings and profit losses, which is in line with the controller requirements that energy cost savings should be maximised, but not exceeded by revenue losses.

7.5.5 Provider Pricing Impact

In this set of experiments we evaluated and compared the performance of the algorithms for different pricing models in order to investigate the impact of the pricing model on the savings from using the proposed approach. Fig. 7.15 presents the results for the CloudSigma and ElasticHosts cloud providers. As can be seen, higher energy cost savings are possible for the CloudSigma pricing scheme (14%) than for the pricing offered by ElasticHosts (2%). This is because CPU provisioning offered by CloudSigma is charged at a lower price resulting in service revenue

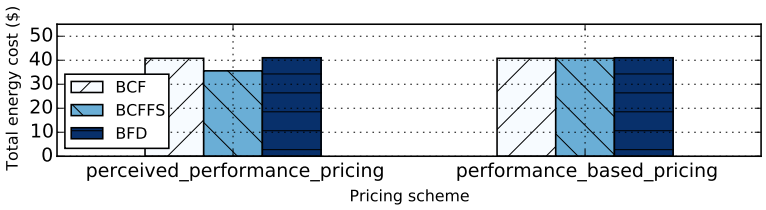


Figure 7.16: Energy cost savings for perceived-performance and performance-based pricing.

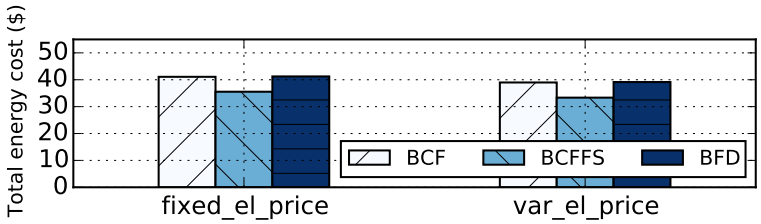


Figure 7.17: Energy cost savings for fixed and variable electricity prices.

being closer to the energy costs. As a result, energy savings gain comparably more weight in the revenue-energy balancing performed by the cloud controller. Since our method applies better to cloud providers like CloudSigma, we used their pricing scheme in all the other simulation scenarios.

7.5.6 Pricing Model Impact

In this experiment we compared the savings obtained by using different pricing models. These include the perceived-performance pricing model proposed in Section 7.3.3 and performance-based pricing offered by the current providers. The results are presented in Fig. 7.16. It can be seen that using performance-based pricing does not lead to energy savings, as the reduction in prices is high compared with the energy costs. As a result, CPU frequency scaling is not feasible. On the other hand, using perceived-performance pricing, savings are possible as CPU frequency reduction does not lead to substantially reduced service revenues.

7.5.7 Electricity Cost Variation

As not all cloud providers may have access to real-time electricity pricing, in this set of experiments we evaluate the performance of the proposed controller under fixed electricity pricing. In Fig. 7.17 scenarios for fixed and variable electricity prices are compared to investigate the impact of electricity pricing on the energy savings obtained using the BCFFS controller. The BCFFS controller achieves better performance under variable electricity pricing reducing the energy costs by exploiting runtime information and adapting the cloud configuration according to the electricity price changes within the day. However, cost savings of 10% (compared to the BFD baseline) that are still significant are achieved for the fixed electricity cost scenario.

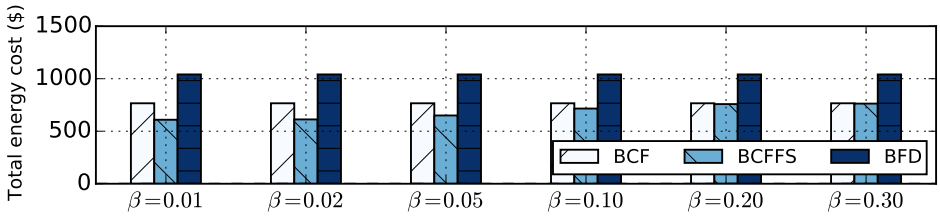


Figure 7.18: Energy cost savings for VMs with different fixed CPU-boundedness.

7.5.8 Variation of Parameter β

Fig. 7.18 shows the results for VMs with fixed CPU-boundedness properties. The aim is to evaluate the impact of different workloads on energy cost savings under the proposed controller and identify workload types where our approach is more beneficial. To do so, simulations using the same set of PMs and VM requests were used, while VM CPU-boundedness properties were varied between 0.0 to 0.4. The results are omitted for larger values of β, where savings are limited due to the impact on application performance. The energy savings achieved by the BCFFS controller decrease gradually while approaching higher values of CPU-boundedness (β). Between a β of 0.0 and 0.2 there is a substantial increase in energy cost as even a small reduction in frequency results in high energy cost savings that exceed the revenue losses. For higher values of β, the savings are limited due to the impact on application performance. As a result, the BCFFS controller achieves the best results for I/O-bound workloads where application performance is not greatly affected by the reduction in frequency.

7.6 Summary

As the demand for cloud platforms increases and as the workload becomes more diverse, a one-fit-all pricing policy does not only provide poor flexibility to the user, but is also not energy efficient. To keep up with the rapid evolution in information infrastructure, a more flexible way of controlling cloud systems must be provided to both satisfy the user and minimize the energy costs.

We have presented a flexible cloud control approach capable of system-level resource management to fit the performance guarantees requested by the user and minimise energy waste by scaling CPU resources on demand. Our cloud controller is driven by a model which covers realistic aspects of real-world cloud platforms. Geotemporal inputs such as real-time electricity pricing and temperature-dependent cooling affecting a geographically distributed cloud provider have been modelled together with a multi-architecture, multi-core power model based on real experiments and used in the Philharmonic cloud simulator to estimate operational costs and the VM service revenue. Several scenarios were examined and two baseline methods were used resulting in energy cost savings of up to 14.57% for ARM and up to 10.06% for Intel architectures.

The lessons learned from our research can be applied to cut costs in data centers. For example, a cloud provider with an $12M annual electricity bill providing VMs on ARM infrastructure at prices similar to CloudSigma for mostly I/O intensive workloads can save around $1.7M,

assuming no frequency scaling was previously used. Even if not all ideal factors are satisfied, e.g. the VM prices are in the ElasticHosts range, around $750k savings can be achieved for a larger cloud provider with a $38M annual energy bill (estimated in [155]).

Our results show that energy costs can be significantly reduced without noticeably impacting the service revenue by scaling the CPU frequencies of the PMs according to the hosted VM characteristics. We have shown that this method applies better to some cloud providers like CloudSigma, where service revenue is closer to energy costs. Savings can be achieved for fixed electricity pricing, but RTEP pricing allows higher energy savings. For our method, ARM architectures are more suitable than Intel, and more I/O-intensive workloads allow for higher savings than CPU-intensive workloads.

CHAPTER 8

Conclusion

In this chapter we give the concluding remarks on the contributions of this thesis, list the limitations of our research and outline possible future work directions.

8.1 Summary

A major challenge of the popular cloud computing paradigm is the rising energy consumption of data centers hosting its computational resources. Furthermore, modern cloud providers frequently operate multiple geographically distributed data centers which result in dynamically-changing environments, constituting geotemporal inputs such as real-time eletricity prices and temperatures that affect cooling efficiency. Existing research explores cloud management actions like VM migration and CPU frequency scaling, but disregarding the additional challenges of geotemporal inputs. We argue that a pervasive cloud control approach is necessary that considers a wider model – from the low-level infrastructure details, like geotemporal inputs affecting power consumption and electricity price – to the high-level aspects of SLA definition of VM QoS properties and pricing details. In this thesis we have presented a collection of management models and cloud control methods that allow geographically distributed cloud providers to address geotemporal inputs with the goal of reducing energy consumption and costs, while satisfying the QoS requirements of end users.

We firstly provided background information on geotemporal inputs, their origins in deregulated electricity markets and temperature-dependent cooling technology. We surveyed the forecasting methods that analyse historical time series to predict their future behaviour. We evaluated two representative forecasting methods – Theta and AAM on geotemporal inputs relevant in the cloud domain as well as presented their performance on a wider range of time series data from the M3 competition. We also presented a concept that would allow for CO_2e emissions to also be treated as an energy cost, considering the Kyoto protocol emission trading scheme. Under this model, CO_2e emissions can be treated as another geotemporal input. We consider that involving data centers in the trade of credits for emission reduction would help encourage more environmentally friendly computational resource allocation.

As our main contributions, we explored methods for dynamically controlling cloud resources based on geotemporal inputs to improve energy and cost efficiency in geographically distributed data centers. We started with a straightforward method for finding peaks in real-time electricity pricing and pausing managed VMs which was evaluated in a realistic OpenStack deployment to show measurable energy savings. We then presented a broader concept of a pervasive cloud controller where a number of geotemporal inputs, such as RTEP and temperature-dependent cooling can be considered in a forward-compatible optimisation engine to reduce energy costs and preserve QoS. We evaluated the approach in a simulation for which we developed a proof-of-concept controller combining forecast-based planning and a hybrid genetic algorithm with greedy local optimisation. Real traces of temperatures and electricity prices were used as input datasets to show a significant energy cost reduction without noticeable QoS implications. A number of parameters such as geographical data center distributions and forecast data quality were explored to serve as cloud provider guidelines to find conditions fit for pervasive cloud control. Similar promising results were shown on the PM layer where we presented methods for determining the best runtime CPU frequency on multi-core Intel and ARM architectures considering both geotemporal inputs and potential revenue losses in performance-based pricing.

To tackle the side effects of energy-aware cloud control methods where VM properties like availability are affected, we proposed a progressive SLA specification system. Traces from cloud control simulations optimised for geotemporal inputs were used to derive corresponding SLAs with precise VM availability and cost estimation. A separate utility-based user SLA selection simulation using real-world datasets was used to validate the energy saving and customer satisfaction potential. Our results show that diverse cloud providers complement each other. For users that require high performance, like web applications, less invasive methods can be used. Scientific computing use cases, on the other hand, can benefit from lower costs of energy-aware VMs.

Finally, we analysed new pricing policies in cloud computing where VM costs are based on the runtime allocated CPU frequencies. We considered ways to adapt cloud control for such performance-based pricing in the context of workloads with variable degrees of CPU boundedness. We developed a perceived-performance pricing scheme that combines both the CPU frequency and the workload's CPU boundedness. This new pricing scheme supports energy-aware methods that consider geotemporal inputs like our CPU frequency scaling cloud controller. Our results show the importance of considering both the computational and the economical side of cloud computing to successfully reduce energy consumption and cost in the underlying geographically distributed data center infrastructure.

8.2 Limitations

In this section we list the limitations and constraints of the contributions presented in this thesis. The topics listed here were out of scope for our research, but can be considered as possible future work directions:

- As mentioned in the introduction, our work focused on IaaS public clouds and while some portions of our work could also apply to other paradigms, certain intricacies of PaaS clouds,

different computational resources like cloud storage etc. can be explored separately in the context of geotemporal inputs for further energy efficiency improvements.

- A question that remains open is how to extend the pervasive cloud control approach for integrated forecasting of arbitrary time series data, e.g. application-level load predictions or local renewable energy availability without using external information sources such as weather forecast services. We have analysed forecasting methods such as AAM and Theta, but we have not integrated them into our proof-of-concept cloud controller, since domain-specific forecast services can provide better accuracy which proved to be critical for planning VM migrations that result in energy savings.

- The AAM method for forecasting geotemporal inputs performed better in our evaluation, but related work shows that other methods, such as Theta perform better with a shorter history available. Additional tests are necessary to find the history length breakpoints that are decisive for the choice of the most accurate forecasting method.

- CO_2e emission prices in markets such as the EU ETS are currently not economically comparable to electricity prices and even less so to VM prices. As a result, it has no effect on cloud control strategies from a financial sense, as any revenue loss from performance-based pricing outweighs savings in CERs. This is why CO_2e was only theoretically studied in Chapter 3 with no simulated evaluation. In the future, however, if environmental aspects of energy consumption become more important, it is possible that a market with higher CO_2e emission prices will arise. The benefits of accounting for CO_2e emissions in cloud control could then become more significant in practice.

- We were mostly focusing on CPU-bound and I/O-bound workload use cases in our work, often in a black box manner, without a deeper analysis of in-app performance metrics. The details of what types of applications are running inside the VMs, whether they are single-threaded, parallel, memory-intensive or disk-intensive, what caching approaches they have etc. can make a big difference on the allocation of VMs. Analysing a wider variety of applications could give more realistic performance and energy consumption models and allow for higher energy savings in cloud control.

- We were mostly using simulations for cloud controller evaluation. Although we used real data traces to make them more relevant, every simulation model must make certain simplifications of real-life phenomena. Only Chapter 4 evaluates the proposed cloud controller on a real OpenStack cloud deployment. The Philharmonic simulator and other data analysis methods used to evaluate the remaining contributions were based on the experience and data obtained from working with real systems, but it is possible that adjustments would be necessary for practical applications of the presented methods.

8.3 Future Work

Finally, in this section we conclude this thesis with a number of possible future work directions beyond the challenges we addressed:

- A next steps in our progressive SLA specification research could be to examine variable SLA metrics, where the probabilistic model would reflect only the most recent data and explore how users might react to such volatile pricing options. As predictions change based on day-night and seasonal changes, exploring time-changing SLAs in the manner of stocks and bonds to match the volatile geotemporal inputs would be feasible.

- SLA violation detection could also be explored as something to integrate into our progressive SLA specification. Perhaps a combination of performance-based pricing, only considering VM availability and not only CPU performance could be used for this purpose.

- A possible future work direction would also be to investigate approaches where the VM migration cloud controller stage of our BCFFS cloud controller also considers the workload CPU-boundedness characteristics in order to maximise the energy savings from using perceived-performance pricing.

- In addition to VMs, Docker containers are emerging as a popular computation encapsulation method in public IaaS clouds. Docker containers are often used in an architecture of immutable application containers and stateful database containers. Such architectures would enable much more efficient computation migration by simply starting application containers at different data center locations with no downtime. It would be interesting to explore our pervasive cloud control approach in such an environment where network latencies between application and database containers would pose a new challenge to address.

All in all, as cloud computing and future large-scale computing paradigms evolve, especially alongside other complex systems we group under the umbrella of geotemporal inputs, methods for making them more energy efficient and environmentally sustainable will continue to be an interesting and challenging field of research.

Bibliography

[1] Gartner Estimates ICT Industry Accounts for 2 Percent of Global CO2 Emissions, 2007. URL: http://www.gartner.com/newsroom/id/503867.

[2] United States Environmental Protection Agency, eGRID2007 Version 1.1. Technical report, 2008. URL: http://www.thegreengrid.org/Global/Content/white-papers/Carbon_Usage_Effectiveness_White_Paper.

[3] Environment Agency - Phase III 2013 to 2020, 2012. URL: http://www.environment-agency.gov.uk/business/topics/pollution/113457.aspx#What_we_need_you_to_do_now.

[4] EU Emissions Trading System - Department of Energy and Climate Change, 2012. URL: http://www.decc.gov.uk/en/content/cms/emissions/eu_ets/eu_ets.aspx.

[5] EU ETS 2009 Company Ratings, 2012. URL: http://www.carbonmarketdata.com/cmd/publications/EU%20ETS%202009%20Company%20Rankings%20-%20June%202010.pdf.

[6] Google's PPAs: What, How and Why, 2012. URL: http://static.googleusercontent.com/external_content/untrusted_dlcp/www.google.com/en/us/green/pdfs/renewable-energy.pdf.

[7] The Green Grid, 2012. URL: http://www.thegreengrid.org/.

[8] The Green Grid – CUE and WUE, 2012. URL: http://www.thegreengrid.org/Global/Content/TechnicalForumPresentation/2011TechForumCUEandWUE.

[9] The Green Grid – The Green Grid Metrics: Describing Data Center Power Efficiency, 2012. URL: http://www.thegreengrid.org/en/Global/Content/white-papers/Green-Grid-Metrics.

[10] IPCC - Intergovernmental Panel on Climate Change, 2012. URL: http://www.ipcc.ch/publications_and_data/publications_ipcc_fourth_assessment_report_synthesis_report.htm.

[11] Kyoto Protocol to the United Nations Framework Convention on Climate Change, 2012. URL: http://unfccc.int/essential_background/kyoto_protocol/items/1678.php.

[12] Planetlab workload traces, 2012. URL: https://github.com/beloglazov/planetlab-workload-traces.

[13] Understanding Energy Reduction Assets (ERAs). Technical report, Joule Assets, 2012. URL: http://jouleassets.com/texteras.

[14] Gartner Says Worldwide Public Cloud Services Market to Total $131 Billion, 2013. URL: http://www.gartner.com/newsroom/id/2352816.

[15] GWA-T-2 Grid5000, 2014. URL: http://gwa.ewi.tudelft.nl/datasets/gwa-t-2-grid5000.

[16] Berserk, 2015. URL: https://github.com/philharmonic/berserk.

[17] Cloudsigma, 2015. URL: https://www.cloudsigma.com/.

[18] Elastichosts, 2015. URL: http://www.elastichosts.co.uk/.

[19] Forecast, 2015. URL: http://forecast.io/.

[20] AccuWeather.com, 2016. URL: http://www.accuweather.com/.

[21] Amazon Elastic Compute Cloud (EC2), 2016. URL: https://aws.amazon.com/ec2/.

[22] Ameren - Historical electricity price data, 2016. URL: https://www2.ameren.com/RetailEnergy/rtpDownload.aspx.

[23] Data Center Energy Efficiency Calculator, 2016. URL: http://www.42u.com/efficiency/energy-efficiency-calculator.htm.

[24] ENERGY STAR Data Center Energy Efficiency Initiatives, 2016. URL: http://www.energystar.gov/index.cfm?c=prod_development.server_efficiency.

[25] Gartner Says Worldwide Public Cloud Services Market Is Forecast to Reach $204 Billion in 2016, 2016. URL: http://www.gartner.com/newsroom/id/3188817.

[26] Google Compute Engine, 2016. URL: https://cloud.google.com/compute/.

[27] Z. Abbasi, T. Mukherjee, G. Varsamopoulos, and S. K S Gupta. Dynamic hosting management of web based applications over clouds. In *2011 18th International Conference on High Performance Computing (HiPC)*, pages 1–10, December 2011. doi:10.1109/HiPC.2011.6152731.

[28] Shah Agam. Lenovo building its first prototype ARM server. February 2015. URL: http://www.computerworld.com/article/2885164/lenovo-building-its-first-prototype-arm-server.html.

[29] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir. Deconstructing Amazon EC2 Spot Instance Pricing. In *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 304 –311, December 2011. doi:10.1109/CloudCom.2011.48.

[30] D. Aikema, C. Kiddle, and R. Simmonds. Energy-cost-aware scheduling of HPC workloads. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pages 1 –7, June 2011. doi:10.1109/WoWMoM.2011.5986476.

[31] D. Aikema, A. Mirtchovski, C. Kiddle, and R. Simmonds. Green cloud VM migration: Power use analysis. In *Green Computing Conference (IGCC), 2012 International*, pages 1–6, June 2012. doi:10.1109/IGCC.2012.6322249.

[32] M. H. Albadi and E.F. El-Saadany. Demand Response in Electricity Markets: An Overview. In *IEEE Power Engineering Society General Meeting, 2007*, pages 1–5, 2007. doi:10.1109/PES.2007.385728.

[33] Scott Alfeld, Carol Barford, and Paul Barford. Toward an analytic framework for the electrical power grid. In *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*, e-Energy '12, pages 9:1–9:4, New York, NY, USA, 2012. ACM. URL: http://doi.acm.org/10.1145/2208828.2208837, doi:10.1145/2208828.2208837.

[34] A. Alnowiser, E. Aldhahri, A. Alahmadi, and M.M. Zhu. Enhanced weighted round robin (EWRR) with DVFS technology in cloud. In *Computational Science and Computational Intelligence (CSCI), 2014 International Conference on*, volume 1, pages 320–326, March 2014. doi:10.1109/CSCI.2014.62.

[35] P. Alonso, R.M. Badia, J. Labarta, M. Barreda, M.F. Dolz, R. Mayo, E.S. Quintana-Orti, and R. Reyes. Tools for Power-Energy Modelling and Analysis of Parallel Scientific Applications. In *2012 41st International Conference on Parallel Processing (ICPP)*, pages 420 –429, September 2012. doi:10.1109/ICPP.2012.57.

[36] David Andrews. The potential contribution of emergency diesel standby generators in dealing with the variability of renewable energy sources. *Renewable electricity and the grid: the challenge of variability*, pages 143–150, 2008.

[37] John Asker and Estelle Cantillon. Properties of scoring auctions. *The RAND Journal of Economics*, 39(1):69–85, 2008.

[38] V. Assimakopoulos and K. Nikolopoulos. The theta model: a decomposition approach to forecasting. *International Journal of Forecasting*, 16(4):521–530, October 2000. doi:10.1016/S0169-2070(00)00066-2.

[39] Luiz André Barroso and Urs Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture*, 4(1):1–108, 2009. URL: http://www.morganclaypool.com/doi/abs/10.2200/s00193ed1v01y200905cac006.

[40] Thomas Beauvisage. Computer usage in daily life. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 575–584, New York, NY, USA, 2009. ACM. URL: http://doi.acm.org/10.1145/1518701.1518791, doi:10.1145/1518701.1518791.

[41] Christian Belady. Carbon Usage Effectiveness (CUE): A Green Grid Data Center Sustainability Metric. White Paper, The Green Grid, 2010. URL: http://www.thegreengrid.org/Global/Content/white-papers/Carbon_Usage_Effectiveness_White_Paper.

[42] A. Beloglazov and R. Buyya. Energy Efficient Allocation of Virtual Machines in Cloud Data Centers. In *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)*, pages 577–578, 2010. doi:10.1109/CCGRID.2010.45.

[43] A. Beloglazov and R. Buyya. Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data Centers Under Quality of Service Constraints. *IEEE Transactions on Parallel and Distributed Systems*, 24(7):1366–1379, 2013. doi:10.1109/TPDS.2012.240.

[44] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Generation Computer Systems*, 28(5):755–768, May 2012. URL: http://www.sciencedirect.com/science/article/pii/S0167739X11000689, doi:10.1016/j.future.2011.04.017.

[45] Anton Beloglazov and Rajkumar Buyya. OpenStack Neat: A Framework for Dynamic and Energy-Efficient Consolidation of Virtual Machines in OpenStack Clouds. *Concurrency and Computation: Practice and Experience (CCPE)*, 2014. URL: http://openstack-neat.org/.

[46] Andreas Berl, Erol Gelenbe, Marco Di Girolamo, Giovanni Giuliani, Hermann De Meer, Minh Quan Dang, and Kostas Pentikousis. Energy-efficient cloud computing. *The Computer Journal*, 53(7):1045–1051, 2010. URL: http://comjnl.oxfordjournals.org/content/53/7/1045.short.

[47] Andreas Berl, Gergö Lovász, Ferdinand von Tüllenburg, and Hermann de Meer. Modelling Power Adaption Flexibility of Data Centres for Demand-Response Management. In *Energy Efficiency in Large Scale Distributed Systems*, pages 63–66. Springer, 2013. URL: http://link.springer.com/chapter/10.1007/978-3-642-40517-4_6.

[48] Philipp Berndt and Andreas Maier. Towards Sustainable IaaS Pricing. In Jörn Altmann, Kurt Vanmechelen, and Omer F. Rana, editors, *Economics of Grids, Clouds, Systems, and Services*, number 8193 in Lecture Notes in Computer Science, pages 173–184. Springer International Publishing, January 2013. URL: http://link.springer.com/chapter/10.1007/978-3-319-02414-1_13.

[49] Ricardo Bianchini, Inigo Goiri, Kien Le, and Thu D. Nguyen. Parasol: A Solar-Powered microDatacenter. *EuroSys*, 2012. URL: http://www.cs.rutgers.edu/~ricardob/papers/parasol.pdf.

[50] A.E.H. Bohra and V. Chaudhary. VMeter: Power modelling for virtualized clouds. In *2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pages 1 –8, April 2010. doi:10.1109/IPDPSW.2010.5470907.

[51] Marcel Boiteux. Peak-load pricing. *The Journal of Business*, 33(2):157–179, 1960. URL: http://www.jstor.org/stable/10.2307/2351015.

[52] P. Bonacquisto, G. D. Modica, G. Petralia, and O. Tomarchio. A Procurement Auction Market to Trade Residual Cloud Computing Capacity. *IEEE Transactions on Cloud Computing*, 3(3):345–357, July 2015. doi:10.1109/TCC.2014.2369435.

[53] Ivan Breskovic, Jörn Altmann, and Ivona Brandic. Creating Standardized Products for Electronic Markets. *Future Gener. Comput. Syst.*, 29(4):1000–1011, June 2013. URL: http://dx.doi.org/10.1016/j.future.2012.06.007, doi:10.1016/j.future.2012.06.007.

[54] Ivan Breskovic, Ivona Brandic, and Jorn Altmann. Maximizing Liquidity in Cloud Markets Through Standardization of Computational Resources. In *Proceedings of the 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, SOSE '13, pages 72–83, Washington, DC, USA, 2013. IEEE Computer Society. URL: http://dx.doi.org/10.1109/SOSE.2013.48, doi:10.1109/SOSE.2013.48.

[55] Peter J. Brockwell and Richard A. Davis. *Introduction to time series and forecasting*. Springer Verlag, 2002.

[56] Niv Buchbinder, Navendu Jain, and Ishai Menache. Online Job-Migration for Reducing the Electricity Bill in the Cloud. In Jordi Domingo-Pascual, Pietro Manzoni, Sergio Palazzo, Ana Pont, and Caterina Scoglio, editors, *NETWORKING 2011*, number 6640 in Lecture Notes in Computer Science, pages 172–185. Springer Berlin Heidelberg, January 2011. URL: http://link.springer.com/chapter/10.1007/978-3-642-20757-0_14.

[57] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya. Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications #x2019; QoS. *IEEE Transactions on Cloud Computing*, 3(4):449–458, October 2015. doi:10.1109/TCC.2014.2350475.

[58] Milan De Cauwer and Barry O'Sullivan. A Study of Electricity Price Features on Distributed Internet Data Centers. In Jörn Altmann, Kurt Vanmechelen, and Omer F. Rana, editors, *Economics of Grids, Clouds, Systems, and Services - 10th International Conference, GECON 2013, Zaragoza, Spain, September 18-20, 2013. Proceedings*, volume 8193 of *Lecture Notes in Computer Science*, pages 60–73. Springer, 2013. doi:10.1007/978-3-319-02414-1_5.

[59] Joseph Chabarek, Joel Sommers, Paul Barford, Cristian Estan, David Tsiang, and Steve Wright. Power awareness in network design and routing. In *In Proc. IEEE INFOCOM*, 2008.

[60] Jeffrey S. Chase, Darrell C. Anderson, Prachi N. Thakar, Amin M. Vahdat, and Ronald P. Doyle. Managing energy and server resources in hosting centers. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 103–116, 2001. URL: http://dl.acm.org/citation.cfm?id=502045%C3%DC.

[61] Junliang Chen, Chen Wang, Bing Bing Zhou, Lei Sun, Young Choon Lee, and Albert Y. Zomaya. Tradeoffs Between Profit and Customer Satisfaction for Service Provisioning in the Cloud. In *Proceedings of the 20th International Symposium on High Performance Distributed Computing*, HPDC '11, pages 229–238, New York, NY, USA, 2011. ACM. URL: http://doi.acm.org/10.1145/1996130.1996161, doi:10.1145/1996130.1996161.

[62] Chi Ching Chi, Mauricio Alvarez-Mesa, and Ben Juurlink. Low-power high-efficiency video decoding using general-purpose processors. *ACM Trans. Archit. Code Optim.*, 11(4):56:1–56:25, January 2015. URL: http://doi.acm.org/10.1145/2685551, doi:10.1145/2685551.

[63] Y. J. Chiang, Y. C. Ouyang, and C. H. (Hsu. An Efficient Green Control Algorithm in Cloud Computing for Cost Optimization. *IEEE Transactions on Cloud Computing*, 3(2):145–155, April 2015. doi:10.1109/TCC.2014.2350492.

[64] Sangyeun Cho and R.G. Melhem. On the interplay of parallelization, program performance, and energy consumption. *Parallel and Distributed Systems, IEEE Transactions on*, 21(3):342–353, 2010. doi:10.1109/TPDS.2009.41.

[65] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association. URL: http://dl.acm.org/citation.cfm?id=1251203.1251223.

[66] Fábio Coutinho, Luis Alfredo V. de Carvalho, and Renato Santana. A Workflow Scheduling Algorithm for Optimizing Energy-Efficient Grid Resources Usage. In *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC)*, pages 642–649. IEEE, December 2011. doi:10.1109/DASC.2011.115.

[67] Leandro Fontoura Cupertino, Georges Da Costa, and Jean-Marc Pierson. Towards a generic power estimator. *Computer Science - Research and Development*, pages 1–9, 2014. URL: http://dx.doi.org/10.1007/s00450-014-0264-x, doi:10.1007/s00450-014-0264-x.

[68] Edward Curry, Souleiman Hasan, Mark White, and Hugh Melvin. An Environmental Chargeback for Data Center and Cloud Computing Consumers. In *Energy Efficient Data Centers*, number 7396 in Lecture Notes in Computer Science, pages 117–128. Springer Berlin Heidelberg, January 2012. URL: http://link.springer.com/chapter/10.1007/978-3-642-33645-4_11.

[69] G. Da Costa, J.-P. Gelas, Y. Georgiou, L. Lefevre, A.-C. Orgerie, J.-M. Pierson, O. Richard, and K. Sharma. The GREEN-NET framework: Energy efficiency in large scale distributed systems. In *IEEE International Symposium on Parallel Distributed Processing, 2009. IPDPS 2009*, pages 1 –8, May 2009. doi:10.1109/IPDPS.2009.5160975.

[70] U. Deshpande and K. Keahey. Traffic-Sensitive Live Migration of Virtual Machines. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 51–60, May 2015. doi:10.1109/CCGrid.2015.163.

[71] J. Doyle, R. Shorten, and D. O'Mahony. Stratus: Load Balancing the Cloud for Carbon Emissions Control. *IEEE Transactions on Cloud Computing*, 1(1):1–1, January 2013. doi:10.1109/TCC.2013.4.

[72] Dražen Lučanin. Philharmonic, 2014. URL: https://philharmonic.github.io/.

[73] Dražen Lučanin, Foued Jrad, Ivona Brandić, and Achim Streit. Energy-Aware Cloud Management through Progressive SLA Specification. In *Economics of Grids, Clouds, Systems, and Services - 11th International Conference, GECON 2014, Cardiff, UK, September 16-18, 2014. Proceedings*, 2014.

[74] Bradley Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. CRC Press, May 1994.

[75] Denny Ellerman and Paul Joskow. The European Union's Emissions Trading System in Perspective, Prepared for the Pew Center on Global Climate Change, 2012. URL: http://www.c2es.org/eu-ets.

[76] Maja Etinski, Julita Corbalan, Jesus Labarta, and Mateo Valero. Optimizing job performance under a given power constraint in HPC centers. In *Proceedings of the International Green Computing Conference (IGCC)*, pages 257–267, 2010.

[77] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, pages 13–23, New York, NY, USA, 2007. ACM. URL: http://doi.acm.org/10.1145/1250662.1250665, doi:10.1145/1250662.1250665.

[78] H. M. Fard, R. Prodan, J. J. D. Barrionuevo, and T. Fahringer. A Multi-objective Approach for Workflow Scheduling in Heterogeneous Environments. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pages 300–309, 2012. URL: http://dl.acm.org/citation.cfm?id=2310188.

[79] Eugen Feller, Louis Rilling, and Christine Morin. Snooze: A Scalable and Autonomic Virtual Machine Management Framework for Private Clouds. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012)*, CCGRID '12, pages 482–489, Washington, DC, USA, 2012. IEEE Computer Society. URL: http://dx.doi.org/10.1109/CCGrid.2012.71, doi:10.1109/CCGrid.2012.71.

[80] Christoph Michael Flath. *Flexible Demand in Smart Grids-Modeling and Coordination*. PhD thesis, Karlsruhe, Karlsruhe Institute for Technology (KIT), Diss., 2013, 2013.

[81] I. Foster, Yong Zhao, I. Raicu, and Shiyong Lu. Cloud Computing and Grid Computing 360-Degree Compared. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1–10, 2008. doi:10.1109/GCE.2008.4738445.

[82] Emilio Francesquini, Márcio Castro, H. Penna, Pedro, Fabrice Dupros, Cota De Freitas, Henrique, Philippe Olivier Alexandre Navaux, and Jean-François Mehaut. On the Energy Eciency and Performance of Irregular Application Executions on Multicore, NUMA and Manycore Platforms. *Journal of Parallel and Distributed Computing*, 76:pp. 32–48, February 2015. URL: https://hal-brgm.archives-ouvertes.fr/hal-01092325, doi:10.1016/j.jpdc.2014.11.002.

[83] Vincent W Freeh, David K Lowenthal, Feng Pan, Nandini Kappiah, Robert Springer, Barry L Rountree, and Mark E Femal. Analyzing the energy-time trade-off in high-performance computing applications. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 18(6):835–848, 2007.

[84] Everette S. Gardner and Ed Mckenzie. Forecasting Trends in Time Series. *Management Science*, 31(10):1237–1246, October 1985. doi:10.1287/mnsc.31.10.1237.

[85] Saurabh Kumar Garg, Chee Shin Yeo, Arun Anandasivam, and Rajkumar Buyya. Environment-conscious scheduling of HPC applications on distributed Cloud-oriented data centers. *Journal of Parallel and Distributed Computing*, 71(6):732–749, June 2011. doi:10.1016/j.jpdc.2010.04.004.

[86] Inigo Goiri, William Katsak, Kien Le, Thu D. Nguyen, and Ricardo Bianchini. Parasol and greenswitch: Managing datacenters powered by renewable energy. In *ACM SIGARCH Computer Architecture News*, volume 41, pages 51–64. ACM, 2013. URL: http://dl.acm.org/citation.cfm?id=2451123.

[87] David Edward Goldberg. *Genetic algorithms in search, optimization, and machine learning*, volume 412. Addison-wesley Reading Menlo Park, 1989.

[88] M Grubb, C Vrolijk, and D Brack. *Kyoto Protocol: A Guide and Assessment*. Earthscan Ltd, London, July 1999. URL: http://www.amazon.de/exec/obidos/redirect?tag=citeulike01-21&path=ASIN/1853835811.

[89] Huseyin Guler, Barla Cambazoglu, and Oznur Ozkasap. Cutting Down the Energy Cost of Geographically Distributed Cloud Data Centers. In *Energy Efficiency in Large Scale Distributed Systems*, pages 279–286, Vienna, 2013. Springer Berlin Heidelberg.

[90] R. Hassan and G. Radman. Survey on smart grid. In *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the*, pages 210–213, 2010. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5453886.

[91] Simon Holmbacka, Sébastien Lafond, and Johan Lilius. Performance monitor based power management for big.LITTLE platforms. In Dimitrios Nikolopoulos and Jose-Luis Nunez-Yanez, editors, *Workshop on Energy Efficiency with Heterogeneous Computing*, page 1 – 6. HiPEAC, 2015.

[92] Simon Holmbacka, Erwan Nogues, Maxime Pelcat, Sébastien Lafond, and Johan Lilius. Energy efficiency and performance management of parallel dataflow applications. In Ana Pinzari and Adam Morawiec, editors, *Conference on Design & Architectures for Signal & Image Processing*, page 1 – 8. ECDI Electronic Chips & Systems design initiative, 2014.

[93] H. J. Hong, D. Y. Chen, C. Y. Huang, K. T. Chen, and C. H. Hsu. Placing Virtual Machines to Optimize Cloud Gaming Experience. *IEEE Transactions on Cloud Computing*, 3(1):42–53, January 2015. doi:10.1109/TCC.2014.2338295.

[94] Chung-Hsing Hsu and Ulrich Kremer. The design, implementation, and evaluation of a compiler algorithm for CPU energy reduction. *ACM SIGPLAN Notices*, 38(5):38–48, 2003.

[95] Chih hsuan Hsu, Cho-Chin Lin, and Tsan-Sheng Hsu. Energy-conscious cloud computing adopting DVFS and state-switching for workflow applications. In *Cloud Computing and Big Data (CloudCom-Asia), 2013 International Conference on*, pages 1–8, Dec 2013. doi:10.1109/CLOUDCOM-ASIA.2013.60.

[96] Xiao-Bing Hu, Wen-Hua Chen, and E. Di Paolo. Multiairport Capacity Management: Genetic Algorithm With Receding Horizon. *IEEE Transactions on Intelligent Transportation Systems*, 8(2):254–263, June 2007. doi:10.1109/TITS.2006.890067.

[97] John D. Hunter. Matplotlib: A 2d Graphics Environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4160265, doi:10.1109/MCSE.2007.55.

[98] Rob J. Hyndman and Baki Billah. Unmasking the Theta method. *International Journal of Forecasting*, 19(2):287–290, April 2003. doi:10.1016/S0169-2070(01)00143-1.

[99] Fredric Hällis, Simon Holmbacka, Wictor Lund, Robert Slotte, Sébastien Lafond, and Johan Lilius. Thermal influence on the energy efficiency of workload consolidation in many-core architecture. In Raffaele Bolla, Franco Davoli, Phuoc Tran-Gia, and Tuan Trinh Anh, editors, *Proceedings of the 24th Tyrrhenian International Workshop on Digital Communications*, page 1–6. IEEE, 2013.

[100] S. Ibrahim, Bingsheng He, and Hai Jin. Towards Pay-As-You-Consume Cloud Computing. In *2011 IEEE International Conference on Services Computing (SCC)*, pages 370–377, July 2011. doi:10.1109/SCC.2011.38.

[101] N. Ioannou, M. Kauschke, M. Gries, and M. Cintra. Phase-based application-driven hierarchical power management on the single-chip cloud computer. In *Parallel Architectures and Compilation Techniques (PACT), 2011 International Conference on*, pages 131–142, Oct 2011. doi:10.1109/PACT.2011.19.

[102] H. Jin, X. Wang, S. Wu, S. Di, and X. Shi. Towards Optimized Fine-Grained Pricing of IaaS Cloud Platform. *IEEE Transactions on Cloud Computing*, 3(4):436–448, October 2015. doi:10.1109/TCC.2014.2344680.

[103] Jonathan Koomey. Growth in Data center electricity use 2005 to 2010. Technical report, Analytics Press, Oakland, CA, August 2011. URL: http://www.analyticspress.com/datacenters.html.

[104] Foued Jrad, Jie Tao, Rico Knapper, Christoph M. Flath, and Achim Streit. A utility-based approach for customised cloud service selection. *Int. J. Computational Science and Engineering*, forthcoming.

[105] Pepple Ken. *Deploying OpenStack*. O'Reilly Media, July 2011. URL: http://shop.oreilly.com/product/0636920021674.do.

[106] M. Kesavan, I. Ahmad, O. Krieger, R. Soundararajan, A. Gavrilovska, and K. Schwan. Practical Compute Capacity Management for Virtualized Datacenters. *IEEE Transactions on Cloud Computing*, 1(1):1–1, January 2013. doi:10.1109/TCC.2013.8.

[107] N.S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J.S. Hu, M.J. Irwin, M. Kandemir, and V. Narayanan. Leakage current: Moore's law meets static power. *Computer*, 36(12):68–75, Dec 2003. doi:10.1109/MC.2003.1250885.

[108] Sonja Klingert, Andreas Berl, Michael Beck, Radu Serban, Marco Girolamo, Giovanni Giuliani, Hermann Meer, and Alfons Salden. Sustainable Energy Management in Data Centers through Collaboration. In *Energy Efficient Data Centers*, volume 7396, pages 13–24. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. URL: http://link.springer.com/chapter/10.1007/978-3-642-33645-4_2?null.

[109] R. Knapper, C.M. Flath, B. Blau, A. Sailer, and C. Weinhardt. A multi-attribute service portfolio design problem. In *Service-Oriented Computing and Applications (SOCA), 2011 IEEE International Conference on*, pages 1–7, 2011. doi:10.1109/SOCA.2011.6166207.

[110] Joanna Kolodziej, Samee Ullah Khan, Lizhe Wang, Aleksander Byrski, Nasro Min-Allah, and Sajjad Ahmad Madani. Hierarchical genetic-based grid scheduling with energy optimization. *Cluster Computing*, 16(3):591–609, 2013. URL: http://link.springer.com/article/10.1007/s10586-012-0226-7.

[111] Jonathan G Koomey. Worldwide electricity used in data centers. *Environmental Research Letters*, 3:034008, July 2008. URL: http://iopscience.iop.org/1748-9326/3/3/034008, doi:10.1088/1748-9326/3/3/034008.

[112] Saurabh Kumar and Rajkumar Buyya. *Green Cloud Computing and Environmental Sustainability*, pages 315–339. John Wiley & Sons, Ltd, 2012. URL: http://dx.doi.org/10.1002/9781118305393.ch16, doi:10.1002/9781118305393.ch16.

[113] S. Lamparter, S. Ankolekar, S. Grimm, and R.Studer. Preference-based Selection of Highly Configurable Web Services. In *Proc. of the 16th Int. World Wide Web Conference (WWW'07)*, pages 1013–1022, Banff, Canada, May 2007.

[114] F. Larumbe and B. Sanso. A Tabu Search Algorithm for the Location of Data Centers and Software Components in Green Cloud Computing Networks. *IEEE Transactions on Cloud Computing*, 1(1):22–35, January 2013. doi:10.1109/TCC.2013.2.

[115] Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems*. Society for Industrial and Applied Mathematics, 1987.

[116] Kien Le, Ricardo Bianchini, Jingru Zhang, Yogesh Jaluria, Jiandong Meng, and Thu D. Nguyen. Reducing Electricity Cost Through Virtual Machine Placement in High Performance Computing Clouds. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 22:1–22:12, New York, NY, USA, 2011. ACM. URL: http://doi.acm.org/10.1145/2063384.2063413, doi:10.1145/2063384.2063413.

[117] Jie Li, Zuyi Li, Kui Ren, and Xue Liu. Towards Optimal Electric Demand Management for Internet Data Centers. *IEEE Transactions on Smart Grid*, 3(1):183–192, March 2012. doi:10.1109/TSG.2011.2165567.

[118] K. Li, Q. Zhang, S. Kwong, M. Li, and R. Wang. Stable Matching-Based Selection in Evolutionary Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, 18(6):909–923, December 2014. doi:10.1109/TEVC.2013.2293776.

[119] Wubin Li, J. Tordsson, and E. Elmroth. Modeling for Dynamic Cloud Scheduling Via Migration of Virtual Machines. In *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 163–171, November 2011. doi:10.1109/CloudCom.2011.31.

[120] Minghong Lin, Zhenhua Liu, A. Wierman, and L.L.H. Andrew. Online algorithms for geographical load balancing. In *Green Computing Conference (IGCC), 2012 International*, pages 1–10, June 2012. doi:10.1109/IGCC.2012.6322266.

[121] Haikun Liu, Cheng-Zhong Xu, Hai Jin, Jiayu Gong, and Xiaofei Liao. Performance and energy modeling for live migration of virtual machines. In *Proceedings of the 20th international symposium on High performance distributed computing*, pages 171–182, 2011. URL: http://dl.acm.org/citation.cfm?id=1996154.

[122] J. Liu, Y. Zhang, Y. Zhou, D. Zhang, and H. Liu. Aggressive Resource Provisioning for Ensuring QoS in Virtualized Environments. *IEEE Transactions on Cloud Computing*, 3(2):119–131, April 2015. doi:10.1109/TCC.2014.2353045.

[123] Zhenhua Liu, Yuan Chen, Cullen Bash, Adam Wierman, Daniel Gmach, Zhikui Wang, Manish Marwah, and Chris Hyser. Renewable and cooling aware workload management for sustainable data centers. In *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '12, pages 175–186, New York, NY, USA, 2012. ACM. URL: http://doi.acm.org/10.1145/2254756.2254779, doi:10.1145/2254756.2254779.

[124] Zhenhua Liu, Adam Wierman, Yuan Chen, Benjamin Razon, and Niangjun Chen. Data center demand response: Avoiding the coincident peak via workload shifting and local generation. *Performance Evaluation*, 70(10):770–791, October 2013. URL: http://www.sciencedirect.com/science/article/pii/S0166531613000928, doi:10.1016/j.peva.2013.08.014.

[125] Ilja Livenson and Erwin Laure. Towards transparent integration of heterogeneous cloud storage platforms. In *Proceedings of the fourth international workshop on Data-intensive distributed computing*, DIDC '11, pages 27–34, New York, NY, USA, 2011. ACM. URL: http://doi.acm.org/10.1145/1996014.1996020, doi:10.1145/1996014.1996020.

[126] D. Lucanin and I. Brandic. Pervasive Cloud Controller for Geotemporal Inputs. *IEEE Transactions on Cloud Computing*, PP(99):1–1, 2016. doi:10.1109/TCC.2015.2464794.

[127] D. Lucanin, I. Pietri, I. Brandic, and R. Sakellariou. A Cloud Controller for Performance-Based Pricing. In *2015 IEEE 8th International Conference on Cloud Computing (CLOUD)*, pages 155–162, June 2015. doi:10.1109/CLOUD.2015.30.

[128] D. Lucanin, I. Pietri, S. Holmbacka, I. Brandic, J. Lilius, and R. Sakellariou. Performance-Based Pricing in Multi-Core Geo-Distributed Cloud Computing. *IEEE Transactions on Cloud Computing (under review)*, 2016.

[129] Dražen Lučanin, Foued Jrad, Ivona Brandic, and Achim Streit. Energy-aware cloud management through progressive SLA specification. In *11th International Conference on Economics of Grids, Clouds, Systems, and Services (GECON)*, pages 83–98. Springer, 2014.

[130] Drazen Lucanin, Michael Maurer, Toni Mastelic, and Ivona Brandic. Energy Efficient Service Delivery in Clouds in Compliance with the Kyoto Protocol. In *Energy Efficient*

Data Centers, volume 7396 of *Lecture Notes in Computer Science*, pages 93–104. Springer Berlin / Heidelberg, 2012. DOI: 10.1007/978-3-642-33645-4_9. URL: http://www.springerlink.com/content/f1r83222335t6671/abstract/.

[131] J. Luo, L. Rao, and X. Liu. Spatio-Temporal Load Balancing for Energy Cost Optimization in Distributed Internet Data Centers. *IEEE Transactions on Cloud Computing*, 3(3):387–397, July 2015. doi:10.1109/TCC.2015.2415798.

[132] Dražen Lučanin and Ivona Brandic. Take a break: cloud scheduling optimized for real-time electricity pricing. In *Cloud and Green Computing (CGC), 2013 Third International Conference on*, pages 113–120. IEEE, 2013. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6686017, doi:10.1109/CGC.2013.25.

[133] X. Ma, F. Liu, Y. Qi, X. Wang, L. Li, L. Jiao, M. Yin, and M. Gong. A Multiobjective Evolutionary Algorithm Based on Decision Variable Analyses for Multiobjective Optimization Problems With Large-Scale Variables. *IEEE Transactions on Evolutionary Computation*, 20(2):275–298, April 2016. doi:10.1109/TEVC.2015.2455812.

[134] Spyros Makridakis and Michele Hibon. The M3-Competition: results, conclusions and implications. *International journal of forecasting*, 16(4):451–476, 2000.

[135] L. Mashayekhy, M. M. Nejad, and D. Grosu. Physical Machine Resource Management in Clouds: A Mechanism Design Approach. *IEEE Transactions on Cloud Computing*, 3(3):247–260, July 2015. doi:10.1109/TCC.2014.2369419.

[136] C. Mastroianni, M. Meo, and G. Papuzzo. Probabilistic Consolidation of Virtual Machines in Self-Organizing Cloud Data Centers. *IEEE Transactions on Cloud Computing*, 1(2):215–228, July 2013. doi:10.1109/TCC.2013.17.

[137] M. Maurer, I. Brandic, and R. Sakellariou. Simulating autonomic SLA enactment in clouds using case based reasoning. *Towards a Service-Based Internet*, pages 25–36, 2010.

[138] M. Maurer, I. Brandic, and R. Sakellariou. Enacting SLAs in clouds using rules. *Euro-Par 2011 Parallel Processing*, pages 455–466, 2011.

[139] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[140] David Meisner, Brian T. Gold, and Thomas F. Wenisch. PowerNap: eliminating server idle power. *SIGPLAN Not.*, 44(3):205–216, March 2009. URL: http://doi.acm.org/10.1145/1508284.1508269, doi:10.1145/1508284.1508269.

[141] Francisco Javier Mesa-Martinez, Ehsan K. Ardestani, and Jose Renau. Characterizing processor thermal behavior. *SIGPLAN Not.*, 45(3):193–204, March 2010. URL: http://doi.acm.org/10.1145/1735971.1736043, doi:10.1145/1735971.1736043.

[142] Akihiko Miyoshi, Charles Lefurgy, Eric Van Hensbergen, Ram Rajamony, and Raj Rajkumar. Critical power slope: understanding the runtime effects of frequency scaling. In *Proceedings of the 16th International Conference on Supercomputing (ICS)*, pages 35–44. ACM, 2002.

[143] G. Mélard and J.-M. Pasteels. Automatic ARIMA modeling including interventions, using time series expert software. *International Journal of Forecasting*, 16(4):497–508, October 2000. doi:10.1016/S0169-2070(00)00067-4.

[144] National Climatic Data Center (NCDC). Climate Data Online (CDO), May 2013. Access to NCDC's Climate Data Online (CDO) data archive and map search. URL: http://www.ncdc.noaa.gov/cdo-web/.

[145] Tien-Dung Nguyen, An Thuy Nguyen, Man Doan Nguyen, Mui Van Nguyen, and Eui-Nam Huh. An Improvement of Resource Allocation for Migration Process in Cloud Environment. *The Computer Journal*, 57(2):308–318, 2014. URL: http://comjnl.oxfordjournals.org/content/57/2/308.short.

[146] Patrick O'Connor and Andre Kleyner. *Practical reliability engineering*. John Wiley & Sons, 2011.

[147] E.L. Padoin, L. Lima Pilla, M. Castro, F.Z. Boito, P.O.A. Navaux, and J.-F. Mehaut. Performance/energy trade-off in scientific computing: the case of ARM big.LITTLE and Intel Sandy Bridge. *Computers Digital Techniques, IET*, 9(1):27–35, 2015. doi:10.1049/iet-cdt.2014.0074.

[148] Darshan S. Palasamudram, Ramesh K. Sitaraman, Bhuvan Urgaonkar, and Rahul Urgaonkar. Using batteries to reduce the power costs of internet-scale distributed networks. In *Proceedings of the Third ACM Symposium on Cloud Computing*, page 11, 2012. URL: http://dl.acm.org/citation.cfm?id=2391240.

[149] Fernando Perez and Brian E. Granger. IPython: A System for Interactive Scientific Computing. *Computing in Science & Engineering*, 9(3):21–29, 2007. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4160251, doi:10.1109/MCSE.2007.53.

[150] Ilia Pietri and Rizos Sakellariou. Cost-efficient CPU provisioning for scientific workflows on clouds. In *Proceedings of the 12th International Conference on Economics of Grids, Clouds, Systems and Services*. Springer LNCS, 2015.

[151] F. L. Pires and B. Barán. A Virtual Machine Placement Taxonomy. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 159–168, May 2015. doi:10.1109/CCGrid.2015.15.

[152] M. Polverini, A. Cianfrani, S. Ren, and A. Vasilakos. Thermal-Aware Scheduling of Batch Jobs in Geographically Distributed Data Centers. *IEEE Transactions on Cloud Computing*, Early Access Online, 2013. doi:10.1109/TCC.2013.2295823.

[153] Gilles Privat. Smart Building Functional Architecture, FI.ICT-2011-285135 FINSENY D4.3. Technical report, 2013. URL: http://www.fi-ppp-finseny.eu/wp-content/uploads/2013/04/FINSENY_D4.3_v1.01.pdf.

[154] X. Qiu, J. X. Xu, K. C. Tan, and H. A. Abbass. Adaptive Cross-Generation Differential Evolution Operators for Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, 20(2):232–244, April 2016. doi:10.1109/TEVC.2015.2433672.

[155] Asfandyar Qureshi, Rick Weber, Hari Balakrishnan, John Guttag, and Bruce Maggs. Cutting the electric bill for internet-scale systems. *SIGCOMM Comput. Commun. Rev.*, 39(4):123–134, August 2009. URL: http://doi.acm.org/10.1145/1594977.1592584, doi:10.1145/1594977.1592584.

[156] Bharathwaj Raghunathan, Yatish Turakhia, Siddharth Garg, and Diana Marculescu. Cherry-picking: Exploiting process variations in dark-silicon homogeneous chip multi-processors. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, pages 39–44, March 2013. doi:10.7873/DATE.2013.023.

[157] Nikola Rajovic, Paul M Carpenter, Isaac Gelado, Nikola Puzovic, Adrian Ramirez, and MR Valero. Supercomputing with commodity CPUs: are mobile SoCs ready for HPC? In *High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for*, pages 1–12. IEEE, 2013.

[158] Lei Rao, Xue Liu, Le Xie, and Wenyu Liu. Minimizing Electricity Cost: Optimization of Distributed Internet Data Centers in a Multi-Electricity-Market Environment. In *2010 Proceedings IEEE INFOCOM*, pages 1–9, March 2010. doi:10.1109/INFCOM.2010.5461933.

[159] T. Rauber and G. Runger. Energy-aware execution of fork-join-based task parallelism. In *Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium on*, pages 231–240, 2012. doi:10.1109/MASCOTS.2012.35.

[160] Shaolei Ren, Yuxiong He, and Fei Xu. Provably-Efficient Job Scheduling for Energy and Fairness in Geographically Distributed Data Centers. In *2012 IEEE 32nd International Conference on Distributed Computing Systems (ICDCS)*, pages 22 –31, June 2012. doi:10.1109/ICDCS.2012.77.

[161] S. W. Roberts. Control Chart Tests Based on Geometric Moving Averages. *Technometrics*, 1(3):239–250, 1959. ArticleType: research-article / Full publication date: Aug., 1959 / Copyright © 1959 American Statistical Association and American Society for Quality. URL: http://www.jstor.org/stable/1266443, doi:10.2307/1266443.

[162] H. Sasaki, S. Imamura, and K. Inoue. Coordinated power-performance optimization in manycores. In *Parallel Architectures and Compilation Techniques (PACT), 2013 22nd International Conference on*, pages 51–61, 2013. doi:10.1109/PACT.2013.6618803.

[163] F. C. Schweppe, R. D. Tabors, M. C. Caraminis, and R. E. Bohn. *Spot pricing of electricity*. Kluwer Academic Publishers,Norwell, MA, January 1988.

[164] V. Seeker, P. Petoumenos, H. Leather, and B. Franke. Measuring QoE of interactive workloads and characterising frequency governors on mobile devices. In *Workload Characterization (IISWC), 2014 IEEE International Symposium on*, pages 61–70, Oct 2014. doi:10.1109/IISWC.2014.6983040.

[165] Yakun Sophia Shao and David Brooks. Energy characterization and instruction-level energy model of intel's xeon phi processor. In *Proceedings of the 2013 International Symposium on Low Power Electronics and Design*, ISLPED '13, pages 389–394, Piscataway, NJ, USA, 2013. IEEE Press. URL: http://dl.acm.org/citation.cfm?id=2648668.2648758.

[166] Hao Shen, Jun Lu, and Qinru Qiu. Learning based DVFS for simultaneous temperature, performance and energy management. In *Quality Electronic Design (ISQED), 2012 13th International Symposium on*, pages 747–754, March 2012. doi:10.1109/ISQED.2012.6187575.

[167] Weiming Shi and Bo Hong. Towards profitable virtual machine placement in the data center. In *Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing*, pages 138–145. IEEE, 2011.

[168] V. Spiliopoulos, S. Kaxiras, and G. Keramidas. Green governors: A framework for continuously adaptive DVFS. In *Green Computing Conference and Workshops (IGCC), 2011 International*, pages 1–8, July 2011. doi:10.1109/IGCC.2011.6008552.

[169] Zhuo Tang, Zhenzhen Cheng, Kenli Li, and Keqin Li. An efficient energy scheduling algorithm for workflow tasks in hybrids and DVFS-enabled cloud environment. In *Parallel Architectures, Algorithms and Programming (PAAP), 2014 Sixth International Symposium on*, pages 255–261, July 2014. doi:10.1109/PAAP.2014.33.

[170] Pawan Kumar Tiwari and Deo Prakash Vidyarthi. Improved auto control ant colony optimization using lazy ant approach for grid scheduling problem. *Future Generation Computer Systems*, 60:78–89, July 2016. URL: http://www.sciencedirect.com/science/article/pii/S0167739X16300012, doi:10.1016/j.future.2016.01.017.

[171] A. N. Toosi, K. Vanmechelen, K. Ramamohanarao, and R. Buyya. Revenue Maximization with Optimal Capacity Control in Infrastructure as a Service Cloud Markets. *IEEE Transactions on Cloud Computing*, 3(3):261–274, July 2015. doi:10.1109/TCC.2014.2382119.

[172] B.M. Tudor and Yong-Meng Teo. Towards modelling parallelism and energy performance of multicore systems. In *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 2526–2529, 2012. doi:10.1109/IPDPSW.2012.318.

[173] Guido Urdaneta, Guillaume Pierre, and Maarten van Steen. Wikipedia Workload Analysis for Decentralized Hosting. *Elsevier Computer Networks*, 53(11):1830–1845, July 2009. URL: http://www.globule.org/publi/WWADH_comnet2009.html.

[174] Cristiano C. A. Vieira, Luiz F. Bittencourt, and Edmundo R. M. Madeira. Towards a PaaS Architecture for Resource Allocation in IaaS Providers Considering Different Charging Models. In Jörn Altmann, Kurt Vanmechelen, and Omer F. Rana, editors, *Economics of Grids, Clouds, Systems, and Services*, number 8193 in Lecture Notes in Computer Science, pages 185–196. Springer International Publishing, January 2013. URL: http://link.springer.com/chapter/10.1007/978-3-319-02414-1_14.

[175] Gregor Von Laszewski, Lizhe Wang, Andrew J Younge, and Xi He. Power-aware scheduling of virtual machines in DVFS-enabled clusters. In *Proceedings of the IEEE International Conference on Cluster Computing and Workshops (CLUSTER)*, pages 1–10. IEEE, 2009.

[176] Zhibo Wang and Yan-Qing Zhang. Energy-Efficient Task Scheduling Algorithms with Human Intelligence Based Task Shuffling and Task Relocation. In *IEEE/ACM International Conference on Green Computing and Communications (GreenCom), 2011*, pages 38–43. IEEE, August 2011. doi:10.1109/GreenCom.2011.15.

[177] Christoph Weber. *Uncertainty in the Electric Power Industry: Methods and Models for Decision Support*. Springer, 1 edition, October 2004.

[178] Rafal Weron. *Modeling and Forecasting Electricity Loads and Prices: A Statistical Approach*. Wiley, 1 edition, December 2006.

[179] Chia-Ming Wu, Ruay-Shiung Chang, and Hsin-Yu Chan. A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters. *Future Generation Computer Systems (FGCS)*, 37:141–147, 2014.

[180] Hong Xu, Chen Feng, and Baochun Li. Temperature aware workload management in geo-distributed datacenters. In *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*, volume 41, pages 373–374. ACM, 2013. URL: http://dl.acm.org/citation.cfm?id=2465539.

[181] Xu Yang, Zhou Zhou, Sean Wallace, Zhiling Lan, Wei Tang, Susan Coghlan, and Michael E. Papka. Integrating Dynamic Pricing of Electricity into Energy Aware Scheduling for HPC Systems. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, pages 60:1–60:11, New York, NY, USA, 2013. ACM. URL: http://doi.acm.org/10.1145/2503210.2503264, doi:10.1145/2503210.2503264.

[182] L. Yin, J. Sun, L. Zhao, C. Cui, J. Xiao, and C. Yu. Joint Scheduling of Data and Computation in Geo-Distributed Cloud Systems. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 657–666, May 2015. doi:10.1109/CCGrid.2015.83.

[183] J. Yu, M. Kirley, and R. Buyya. Multi-objective planning for workflow execution on Grids. In *Proceedings of the 8th IEEE/ACM International conference on Grid Computing*, pages 10–17, 2007. URL: http://dl.acm.org/citation.cfm?id=1513496.

[184] Qi Zhang, Quanyan Zhu, and R. Boutaba. Dynamic Resource Allocation for Spot Markets in Cloud Computing Environments. In *2011 Fourth IEEE International Conference on Utility and Cloud Computing (UCC)*, pages 178–185, December 2011. doi:10.1109/UCC.2011.33.

[185] Q. Zheng, R. Li, X. Li, and J. Wu. A Multi-objective Biogeography-Based Optimization for Virtual Machine Placement. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 687–696, May 2015. doi:10.1109/CCGrid.2015.25.

[186] Rongliang Zhou, Zhikui Wang, Alan McReynolds, Cullen E. Bash, Thomas W. Christian, and Rocky Shih. Optimization and control of cooling microgrids for data centers. In *Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), 2012 13th IEEE Intersociety Conference on*, pages 338–343. IEEE, 2012. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6231449.

[187] Xiaomin Zhu, L.T. Yang, Huangke Chen, Ji Wang, Shu Yin, and Xiaocheng Liu. Real-Time Tasks Oriented Energy-Aware Scheduling in Virtualized Clouds. *IEEE Transactions on Cloud Computing*, 2(2):168–180, April 2014. doi:10.1109/TCC.2014.2310452.

APPENDIX A

Curriculum Vitæ

Dražen Lučanin

Education

2011–2016	Computer Science PhD, Faculty of Informatics, Vienna University of Technology
2009–2011	Computer Science MSc, Faculty of Electrical Engineering and Computing, University of Zagreb
2006–2009	Computer Science BSc, Faculty of Electrical Engineering and Computing, University of Zagreb

Employment

2015–	Software developer, Dražen Lučanin e.U.
2011–2015	Research assistant, Distributed Systems Group, Vienna University of Technology
2011–2012	External associate, Department of Electronics, Ruđer Bošković Institute, Zagreb
2009–2010	Software developer, Department of Distributed and Multimedia Systems, University of Vienna

Research

Research Projects

- HALEY – Holistic Energy Efficient Approach for the Management of Hybrid Clouds, Vienna University of Technology research award, 2011–2015
- FOC-II – Forecasting Financial Crises, FP7, 2011–2012
- AMASL – Ambient Assisted Shared Living, University of Vienna, 2009–2010

Publications

Refereed Publications in Journals

- Dražen Lučanin, Ivona Brandić. *Pervasive Cloud Controller for Geotemporal Inputs.* IEEE Transactions on Cloud Computing, 2016. doi:10.1109/TCC.2015.2464794
- Dražen Lučanin*, Ilia Pietri*, Simon Holmbacka*, Ivona Brandic, Johan Lilius, Rizos Sakellariou. *Performance-Based Pricing in Multi-Core Geo-Distributed Cloud Computing.* IEEE Transactions on Cloud Computing (under review, *equal contribution).

Refereed Publications in Conference Proceedings

- Soodeh Farokhi, Pooyan Jamshidi, Dražen Lučanin, Ivona Brandić. *Performance-based Vertical Memory Elasticity.* 12th IEEE International Conference on Autonomic Computing (ICAC 2015), 7–10 July, 2015, Gronoble, France. doi:10.1109/ICAC.2015.51
- Dražen Lučanin, Ilia Pietri, Ivona Brandić, Rizos Sakellariou. *A Cloud Controller for Performance-Based Pricing.* 8th IEEE International Conference on Cloud Computing (CLOUD 2015), 27 June – 2 July, 2015, New York, USA. doi:10.1109/CLOUD.2015.30
- Dražen Lučanin, Foued Jrad, Ivona Brandić, and Achim Streit. *Energy-Aware Cloud Management through Progressive SLA Specification.* 11th International Conference Economics of Grids, Clouds, Systems, and Services (GECON 2014). 16–18 September, 2014, Cardiff, UK. doi:10.1007/978-3-319-14609-6_6
- Dražen Lučanin, Ivona Brandić. *Take a break: cloud scheduling optimized for real-time electricity pricing.* Proceedings of the 3rd International Conference on Cloud and Green Computing (CGC), 30 September – 2 October, 2013, Karlsruhe, Germany. doi:10.1109/CGC.2013.25
- Dragan Gamberer, Dražen Lučanin, Tomislav Šmuc. *Analysis of World Bank Indicators for Countries with Banking Crises by Subgroup Discovery Induction.* Proceedings of the 36th International Convention on Information and Communication Technology, Electronics

and Microelectronics – MIPRO. 20–24 May, 2013, Opatija, Croatia. doi:10.1007/978-3-642-33492-4_8

- Toni Mastelić, Dražen Lučanin, Andreas Ipp, Ivona Brandić. *Methodology for trade-off analysis when moving scientific applications to the Cloud.* CloudCom 2012, 4th IEEE International Conference on Cloud Computing Technology and Science. 3–6 December, 2012, Tapei, Taiwan. doi:10.1109/CloudCom.2012.6427575

- Dragan Gamberer, Dražen Lučanin, Tomislav Šmuc. *Descriptive modeling of systemic banking crises.* The 15th International Conference on Discovery Science (DS 2012), 29–31 October, 2012, Lyon, France. doi:10.1007/978-3-642-33492-4_8

- Matija Gulić, Dražen Lučanin, Nina Skorin-Kapov. *A Two-Phase Vehicle based Decomposition Algorithm for Large-Scale Capacitated Vehicle Routing with Time Windows.* Proceedings of the 35th International Convention on Information and Communication Technology, Electronics and Microelectronics – MIPRO, 21–25 May, 2012, Opatija, Croatia.

- Dražen Lučanin, Ivan Fabek, Domagoj Jakobović. *A visual programming language for drawing and executing flowcharts.* Proceedings of the 34th International Convention on Information and Communication Technology, Electronics and Microelectronics – MIPRO. 23–27 May, 2011, Opatija, Croatia. **Best student paper award.**

- Matija Gulić, Dražen Lučanin, Ante Šimić, Šandor Dembitz. *A digit and spelling speech recognition system for the Croatian language.* Proceedings of the 34th International Convention on Information and Communication Technology, Electronics and Microelectronics – MIPRO. 23–27 May, 2011, Opatija, Croatia.

Refereed Publications in Workshop Proceedings

- Dražen Lučanin, Michael Maurer, Toni Mastelić, Ivona Brandić. *Energy Efficient Service Delivery in Clouds in Compliance with the Kyoto Protocol.* 1st International Workshop on Energy-Efficient Data Centers, 8th May, 2012, Madrid, Spain.

Theses

- Dražen Lučanin. *Energy Efficient Cloud Control and Pricing in Geographically Distributed Data Centers.* PhD thesis, Faculty of Informatics, Vienna University of Technology, June, 2016.

- Dražen Lučanin. *Visual definition of procedures for automatic virtual scene generation.* Master's thesis, Faculty of Electrical Engineering and Computing, University of Zagreb, June, 2011.

- Dražen Lučanin. *Graphical user interface for a video-conferencing application accessible to the elderly.* Bachelor's thesis, Faculty of Electrical Engineering and Computing, University of Zagreb, June, 2009.

Activities

Research Visits

- Short Term Scientific Mission (STSM), *Cloud Control for Performance-Based Pricing.* COST Action IC1305 Network for Sustainable Ultrascale Computing (NESUS), carried out at the University of Manchester, Manchester, United Kingdom; 5–20 December, 2014.

- 2nd COST IC804 Training School on Energy Efficiency in Large Scale Distributed Systems, University of Balearic Islands (UIB), Palma de Mallorca, Spain; 24–27 April, 2012.

Scientific Talks

- *Energy-Aware Cloud Management through Progressive SLA Specification.* COST IC1305 NESUS Meeting, Paris, France; 1–2 December, 2014.

- *Energy-aware Cloud Management through Progressive SLA Specification.* 5th Cloud Control Workshop, Moelle, Sweden; 20–22 August, 2014.

- *Collaborations with Univ. Toulouse.* Energy Efficiency in Large Scale Distributed Systems conference, Vienna, Austria; 22–24 April 2013.

- *Kyoto Protocol Compliant Management of Data Centers.* COST 804 focus group "Energy and QoS-aware Workload Manangent in Clouds" meeting, INRIA Rennes – Bretagne Atlantique, France; 27th March, 2012.

Additionally, personally gave talks for all the conference and workshop publications with first authorship.

Other Activities

- Track chair for GECON 2015, 12th International Conference on Economics of Grids, Clouds, Systems and Services.

- Program comitee member for CloudComp 2012, 3rd International Conference on Cloud Computing, Vienna, Austria; 24–26 September, 2012.

- Reviewer for *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Cloud Computing*, *SuperComputing*, *CCGrid*, *(EC)2*, *Euro-Par*, *PDP*, *International Conference on Runtime Verification*, *WETICE*, *WORKS*, *BISE / WIRTSCHAFTSINFORMATIK*.

- Member, IEEE, 2012–2015.

Honours, Awards & Scholarships

- EuroPython attendance grant, 2013.
- Google student grant for attending the EuroPython conference, 2012.
- Best Student Paper award – *A visual programming language for drawing and executing flowcharts.* MIPRO 2011. (see section Research)
- Best Student Computer Program award – *RoboDJ* , Faculty of Electrical Engineering and Computing, 2010.
- Two times Croatian Ministry of Education student scholarship, 2006–2009 and 2009–2011.
- Two times Erasmus student exchange scholarship, 2009 and 2011 at the University of Vienna.

Curriculum vitæ last updated: 19th April, 2016