Energy-Efficient Data Centres: Model-Based Analysis of Power-Performance Trade-Offs

Björn F. Postema

Energy-Efficient Data Centres: Model-Based Analysis of Power-Performance Trade-Offs¹

Björn Frits Postema

¹In het Nederlands vertaald:
“Energie-Efficiënte Datacentra: Model-Gebaseerde Analyse van Vermogen-Prestatie Trade-Offs.”

Graduation committee:

Chairman: Prof. dr. J.N. Kok
Promoter: Prof. dr. ir. B.R.H.M. Haverkort

Members:

Prof. dr. ir. G.J.M. Smit	University of Twente
Prof. dr. J.L. van den Berg	University of Twente
Prof. dr. A.K.I. Remke	Universität Münster
Prof. dr. H. de Meer	Universität Passau
Prof. dr. K. Wolter	Freie Universität Berlin
Prof. dr. ir. A. Iosup	Free University of Amsterdam

UNIVERSITY
OF TWENTE. | DIGITAL SOCIETY
INSTITUTE

DSI Ph.D.-thesis Series No. 18-20

Digital Society Institute
University of Twente
P.O. Box 217, NL – 7500 AE Enschede

Netherlands Organisation
for Scientific Research

This work is part of the research programme “Robust Design of Cyber-Physical Systems” (CPS) with Cooperative Networked Systems (CNS) project number 12696, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO).

ISBN 978-90-365-4688-1

ISSN 2589-7721 (DSI Ph.D.-thesis Series No. 18-20)

DOI 10.3990/1.9789036546881

<https://doi.org/10.3990/1.9789036546881>

Typeset with $\text{\LaTeX}$. Printed by Ipskamp Printing.
Cover design: Roos Jonkheer-Vos

Copyright © 2018 Björn F. Postema. Some Rights Reserved.

This work is licensed under a Creative Commons “Attribution-NonCommercial-ShareAlike 3.0 Unported” licence.

**ENERGY-EFFICIENT DATA CENTRES:
MODEL-BASED ANALYSIS OF
POWER-PERFORMANCE TRADE-OFFS**

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
Prof. dr. T.T.M. Palstra,
volgens besluit van het College voor Promoties,
in het openbaar te verdedigen
op vrijdag 21 december 2018 om 10.45 uur

door

Björn Frits Postema

geboren op 28 maart 1989
te Elburg, Nederland

Dit proefschrift is goedgekeurd door:
Prof. dr. ir. (promotor) B.R.H.M. Haverkort

Dedicated to Jesus Christ.

Summary

Nowadays businesses, governments and industries rely heavily on ICT solutions. Since these ICT solutions often have high space, security, availability and performance requirements, data centres provide physical locations to facilitate networks of servers for processing and storage purposes of these ICT solutions. The increasing worldwide energy consumption of data centres has a significant impact on the world's ecosystem through an increase in greenhouse gases for the generation of the necessary electricity. This has led to an increased attention in the global political agenda. Moreover, the high energy consumption of data centres has also led to high operational costs with the consequence that even the smallest improvements in currently active systems could significantly ease the financial burden. These reasons have led to a greater need for *energy-efficient* data centres.

Over the last years, many energy saving techniques have already been developed to improve energy-efficiency of data centres. We propose to assist energy saving techniques with meaningful insights obtained from model-based analysis of (electric) power and (system) performance. Therefore, we propose two sets of models, one which allows for a numerical solution, and one simulation based, for the analysis of power and performance in energy-efficient data centres. In both approaches, power is made proportional to the utilisation by means of the *power management* of servers, which is considered one of the key contributors to saving energy in data centres. This raises all kinds of questions, such as: How do we distribute the tasks of servers in the network? When should a server be put to sleep? And when should a server again be woken up? By the correct distribution of tasks and the power management of servers, energy can be saved.

Furthermore, a lot of energy is consumed in the *cooling* of all servers, as this is necessary to maintain the longest life cycle for all IT equipment. By a smart distribution of workload over the servers, i.e., by avoiding so-called hot spots, temperatures in the data centre are more evenly spread, which leads to a lower supply temperature for the cooling systems, i.e., less energy is re-

quired for cooling. In our simulation models, we combine the energy saving techniques of both power management and cooling. We introduce a *simulation framework* in which our models are implemented which allows us to analyse power, performance and thermodynamics of a data centre. In addition, the power, performance and thermal-aware distribution of tasks across heterogeneous servers are taken into account. Simultaneously, the switching of power states (e.g., the waking of servers) via power management and the adjustment of supply temperatures for cooling systems are considered.

Power state switching in servers for power management requires strategic decisions to be made. These decisions are made based on information obtained by sensors and server log files. In this dissertation, a structural way to describe and evaluate *power management strategies* is proposed. The power management strategies allow for the minimisation of energy consumption while maintaining acceptable performance. In most cases the overhead has to be minimised which is caused by idle servers and the overly active switching of power states. However, in some cases power can even be exchanged at the expense of performance leading to so-called *power-performance trade-offs*. These trade-offs provide substantial flexibility for data centre owners regarding design decisions. With our analysis, we are able to grow awareness for such trade-offs and, if possible, control of these trade-offs.

To determine the degree to which the models correspond to reality, our models are *experimentally validated* where possible. For this reason, the simulation models will be calibrated with parameters obtained through *workload modelling* using workload traces from a real data centre. This model calibration allows to discuss the consequences of our workload assumptions through comparison of power and performance. Moreover, a *cross-model validation* will be used to compare power and performance estimates of the same system with two different modelling and analysis techniques. Furthermore, we investigate the possibility of using a cheap, low power and widely supported hardware in the form of a *micro data centre* by comparing it with a real data centre. A micro data centre has been used as an experimental setup for validation of the simulation models.

Samenvatting

Vandaag de dag zijn bedrijven, overheden en industrieën sterk afhankelijk van ICT-oplossingen. Aangezien deze oplossingen vaak hoge eisen stellen aan ruimte, veiligheid, beschikbaarheid en prestaties, leveren datacentra fysieke locaties ter facilitering van servernetwerken voor verwerkings- en opslagdoel-einden van deze ICT-oplossingen. Het wereldwijde energieverbruik van data-centra heeft een significante invloed op het ecosysteem van de wereld door een groei in broeikasgassen die vrijkomen bij het genereren van de benodigde elektriciteit. Dit heeft geleid tot een verhoogde aandacht voor datacentra op de globale politieke agenda. Bovendien heeft het hoge energieverbruik van data-centra ook geleid tot hoge operationele kosten. Als gevolg hiervan kan zelfs de kleinste verbetering in de huidige systemen de financiële last aanzienlijk verlichten. Beide redenen hebben geleid tot een grotere behoefte aan *energie-efficiënte* datacentra.

In de afgelopen jaren zijn al veel energiebesparende technieken ontwikkeld om de energie-efficiëntie van datacentra te vergroten. Wij stellen voor om energiebesparende technieken te ondersteunen met betekenisvolle inzichten verkregen uit model-gebaseerde analyse van (elektrisch) vermogen en (systeem)prestaties. Hiervoor presenteren wij twee modellen, één die een numerieke oplossing mogelijk maakt en één op simulatie gebaseerd, die vermogen en prestaties in energie-efficiënte datacentra kunnen analyseren. In beide benaderingen wordt vermogen proportioneel gemaakt aan het gebruik door middel van het *energiebeheer* van servers. Dit wordt beschouwd als één van de belangrijkste methodes om energie te besparen in datacentra. Dit roept allerlei vragen op, zoals: Hoe distribueren wij de taken over de servers in het netwerk? Wanneer zou een server in slaap moeten worden gebracht? En wanneer zou een server weer moeten ontwaken? Door de juiste verdeling van taken en het energiebeheer van servers kan energie worden bespaard.

Daarnaast wordt er veel energie gebruikt voor de *koeling* van alle servers, aangezien dit nodig is om een lange levensduur van alle IT-apparatuur te bereiken. Door een slimme distributie van de werklasten over de servers,

dat wil zeggen door het vermijden van zogenaamde hotspots, is de temperatuur in het datacentrum meer gelijkmatig. Dit leidt tot een lagere benodigde toevoertemperatuur van de koelsystemen waardoor er minder energie nodig is voor koeling. In onze simulatiemodellen combineren wij de energiebesparende technieken van zowel energiebeheer als koeling. Wij introduceren een *simulatieraamwerk* waarin onze modellen zijn geïmplementeerd, die het ons mogelijk maakt om vermogen, prestatie en thermodynamica van een datacentrum te analyseren. Bovendien wordt een vermogens-, prestatie- en thermisch bewuste taakverdeling over heterogene servers worden beschouwd. Tegelijkertijd wordt het wisselen van vermogenstoestanden (bijvoorbeeld het wakker maken van servers) via energiebeheer en het aanpassen van de toevoertemperatuur van koelsystemen beschouwd.

Het schakelen tussen vermogenstoestanden in servers voor energiebeheer, vraagt om strategische beslissingen. Dergelijke beslissingen worden gemaakt op basis van informatie verkregen door sensoren of de serverlogbestanden. In deze dissertatie wordt een structurele manier voorgesteld om *energiebeheer strategieën* te beschrijven en te evalueren. De energiebeheerstrategieën stellen in staat het energieverbruik te minimaliseren met behoud van acceptabele prestaties. In de meeste gevallen moet de overhead worden geminimaliseerd, die veroorzaakt wordt door inactieve servers en het overactief wisselen van vermogenstoestanden. Echter, in sommige gevallen kan vermogen zelfs worden ingeruild ten koste van prestaties wat leidt tot zogenaamde *vermogen-prestatie trade-offs*. Deze trade-offs leveren een substantiële flexibiliteit voor datacentrumeigenaren met betrekking tot ontwerpbeslissingen. Met onze modelanalyse zijn wij in staat om bewustzijn te laten groeien en, indien mogelijk, controle te geven over deze trade-offs.

Om te kunnen bepalen in welke mate de modellen overeenkomen met de werkelijkheid zijn deze waar mogelijk *experimenteel gevalideerd*. Daarom zullen de simulatiemodellen worden gekalibreerd met parameters verkregen door *werklastmodellering*, waarbij gebruik gemaakt wordt van werklasttraces van een echt datacentrum. Deze kalibratie van het model maakt het mogelijk om de gevolgen van onze werklastaannames te bespreken door het vergelijken van vermogen en prestatie. Bovendien wordt een *wederzijdse modelvalidatie* gebruikt om vermogen- en prestatieschattingen van hetzelfde systeem met twee verschillende modellerings- en analysetechnieken te vergelijken. Daarnaast bestuderen wij de mogelijkheid om goedkope, laag-vermogen- en breed ondersteunde hardware te gebruiken in de vorm van een *micro-datacentrum* door vergelijking met een echt datacentrum. Een micro-datacentrum is gebruikt als experimentele opstelling voor validatie van de simulatiemodellen.

Acknowledgements

In the past few years, I have noticed that a thankful heart is often a cheerful heart as well. So, it is my joy and privilege to be able to mention all those who have been of great significance in my life. The opportunities in a wealthy country, such as the Netherlands, are very rich and passed on, shared and distributed by family, friends and colleagues. I believe it comes with freedom that we should embrace, do good with and multiply.

First of all, I would like to thank *God* for being with me in everything. In each and every situation, You were the peace in my heart and the strength to do it all. You made sure, that I felt You by my side. Especially during the hardest times in my life, You promised me a better future and told me who I really am. I dared to believe You, because You assured me of this hope and identity by the greatest expression of love possible. Because you sent Your beloved Son, *Jesus Christ*, I see Your deep care for all creation. With this You have set the greatest example on which I base my personal work ethic (in which I am still growing). It is teaching me how to also be wildly enthusiastic about and deeply in love with all that You have created.

I would like to thank *Prof.dr.ir. Boudewijn Haverkort*, who taught me valuable lessons from his broad experience in academia. I deeply appreciate your humane, enthusiastic and professional approach. You helped me to understand and work well in the academic world. Thank you providing me with numerous opportunities and constructive feedback. These have helped me to grow as an academic during my time in Twente. Also, *Prof.dr. Anne Remke* thank you for introducing me to this wonderful job opportunity. I would also like to thank the committee members for their valuable feedback and especially for their willingness to travel (from far) for a defence held just before the Christmas break.

I would like to thank *Henk Broekhuizen-Versteeg* and *Hamed Ghasemieh* for accepting to be my paranymphs. Henk and his wife Miranda, have become my best friends during my time in Enschede. With you two, my life here became much more meaningful and joyful. Thank you for accepting me for who I am.

Hamed, thank you for being both a colleague and a friend. I appreciate our conversations about our cultures and religion. Also, the mutual support in each others professional careers fills me with gratitude.

I would like to express my thanks to all colleagues from my research group DESIGN AND ANALYSIS OF COMMUNICATION SYSTEMS, that allowed me to work in an ambitious and friendly atmosphere. Specifically, I would like to thank my office roommates *Marijn*, *Anne*, *Justyna*, and *Jair* for all nice conversations about work and life. You really did brighten up the grey-ish office environment beside my precious plants ;-). *Jair*, thank you for being both my friend and colleague; we both already share so many wonderful memories and our conversations have always been positively inspiring. *Jeanette*, I would like to thank you for being so attentive to how I am doing, helping me in so many ways with travel and finding my way at the UNIVERSITY OF TWENTE. *Marijn*, I appreciate the nice and honest conversations and collaboration we have had. *Ricardo*, thank you for all you have shared with me as a friend. *Mozhdeh* and *Morteza*, thank you for your positive attitude and encouraging words you have both shared so often and for your special appreciation of my gluten-free bread. I also would like to thank the bachelor students *Daniël*, *Nick*, *Paul*, *Mick*, *Rob* and *Bas* with whom I enjoyed working on the micro data centre project.

I would like to thank the external colleagues from my project. Specifically, I would like to thank *Reinder* from BETTERBE. As my former internship supervisor, I thank you for the willingness for our informal cooperation that has brought us both benefit. Also, I would like to thank *Tobias* from the UNIVERSITY OF GRONINGEN for the willingness to form a fruitful cooperation that leads to outcomes with great potential. I appreciate your thoroughness as a control theorist, which (eventually, I admit :-P) leads to useful discussions and nice results. *Freek*, I would like to thank you for taking initiative to perform a comparison between our models that have led to a nice chapter in both our theses. I appreciate your enthusiastic work ethic and swift way of reasoning.

I thank the people I have met during my ten years membership of the NAVIGATORS STUDENTENVERENIGING ENSCHEDE. Specifically, I thank *Alinda* and *Marius* for believing in my God-given potential. I thank *Wietske* for awakening the dream to run and make a difference. I thank *GertJan* and *Jacyntha* for their positive energy and courage. Also, for allowing me to pass on the baton to you both to make a difference with running. I thank *Jorrit* for all the visits in which we shared our latest adventures and brought out the best in each other. I thank *Steven* for the great conversations on work and life during our photography sessions. I thank *Erwin* for being good company in the numerous lunch breaks in the last years. I thank *Benjamin* for his caring attitude. I thank *Jan-Willem*

for the willingness to lend me a sympathetic ear and support me in writing. I thank *Alisa* and *Allan* for their exemplary devotion to God and prayers. I thank *Roos* for the willingness to design an amazing cover for this thesis.

My thanks is also expressed to my roommates *Roelof* and *Sjoerd*. Thank you very much for being in my life and choosing to live with me. Thank you both for the friendship and all the nice conversations and joy we have had; I know we are always looking for the best in and for each other.

I would like to thank my brothers and sister, and their families for sympathising with all that happens in my life, and I am grateful for the many good memories we share. I would like to thank my parents *Frits* and *Janneke Postema* for giving me a good and firm foundation to my life. Rightly so, that when I receive any honour, that you both deserve to be honoured as well. I thank you both for the personal sacrifices made for me and the family to provide in all our needs. Father, I will always remember you saying to me, 'hang in there', the many times I needed it. Thank you for showing me your loyal and enthusiastic work ethic. Thank you for encouraging me with your enthusiasm for innovative technology since childhood. Thank you for your steadfast faith in God. Mother, I would like to thank you for heart-warming love you expressed to me. You taught me how to care with love from the heart. Your love makes every meal you prepare stand out by far from anything I have every tasted. You showed me how to create environment in which one can flourish. You showed me resilience through your loyal deep care for our family that is fuelled by your rock-solid faith in Jesus Christ.

— Björn F. Postema

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Power and Performance Model-Based Approach	4
1.3	Goal & Research Questions	5
1.4	Approaches	6
1.5	Thesis Structure	9
1.5.1	Thesis Outline	9
1.5.2	Contributions	10
1.5.3	Contents and Origin of the Chapters	11
2	Data Centres	13
2.1	The Rise of Data Centres	14
2.1.1	History of Data Centres	14
2.1.2	Data Centre Definition	14
2.1.3	Saving Energy in Data Centres	15
2.2	Data Centre Infrastructure	18
2.2.1	Components	18
2.2.2	Important Data Centre Demands	19
2.3	Power Saving Techniques	21
2.3.1	Advanced Cooling	21
2.3.2	Advanced Power Management	22
2.3.3	Advanced Server Consolidation	26
2.4	Load Balancing	27
2.4.1	Existing Load Balancing Techniques	27
2.4.2	Load Balancing in Energy Saving Data Centres	28
2.5	Modelling and Analysis	29
2.5.1	Linear Models	29
2.5.2	Control-Theoretical Models	30
2.5.3	Markov Chains	30
2.5.4	Petri Nets	31

2.5.5	Modelling and Analysis Approaches Overview	31
2.5.6	Discussion	31
2.6	Conclusions	33
I	Modelling	35
3	Stochastic Petri Net Models: A Numerical Approach	37
3.1	Introduction	38
3.2	Single Server Model	39
3.2.1	Basic model	39
3.2.2	Visualisation of stochastic Petri net models	39
3.2.3	Single server stochastic Petri net model	39
3.2.4	Power-performance trade-off	41
3.2.5	Results	42
3.3	Multi-Server Model	45
3.3.1	Stochastic Petri net model	45
3.3.2	Power-performance trade-off	46
3.3.3	Results	46
3.4	Computational Cost and Scalability	52
3.5	Related Work	53
3.6	Conclusions	55
4	Simulation Models	57
4.1	Introduction	58
4.2	System Description	59
4.3	Data Centre Models	60
4.3.1	Model Overview	61
4.3.2	Server Performance Model	61
4.3.3	Server Power Model	62
4.3.4	Cascade Effect Model	64
4.3.5	Workload	65
4.3.6	Power Management Strategies	65
4.3.7	Power-Performance Metrics	66
4.3.8	Estimation Method	68
4.3.9	Simulation Setup	69
4.3.10	Visualisation	69
4.4	Results	71
4.4.1	Case Study: Computational Cluster	71

4.4.2	Cross-Model Validation	73
4.5	Conclusions	77
5	Specification of Advanced Power Management Strategies	79
5.1	Introduction	80
5.2	Overall Approach	80
5.2.1	Observable Quantities	81
5.2.2	Controllable Parts	82
5.2.3	Derived Quantities	83
5.3	Strategy specification	84
5.3.1	Three-Step Approach	84
5.3.2	Satisfiers	86
5.3.3	Constraints	87
5.3.4	Illustrative Example	88
5.4	Application	88
5.4.1	Base Case Strategy: AlwaysOn	90
5.4.2	Literature Inspired Strategies: Optimal and Demotion	90
5.4.3	Fine Tuned Strategies: Strong and Advanced	92
5.5	Results	94
5.5.1	Data Centre Configuration	94
5.5.2	Example Strategies	96
5.6	Power Management Module Implementation	97
5.7	Conclusions	99
6	Evaluation of Advanced Power Management Strategies	101
6.1	Introduction	102
6.2	Overall Approach	102
6.2.1	Chapter-Wide Job and Data Centre Configuration	103
6.2.2	Example Strategy	104
6.3	Strategy Qualities	104
6.3.1	Efficiency	105
6.3.2	Stability	106
6.3.3	Robustness	107
6.3.4	Adaptability	108
6.3.5	Discussion	110
6.4	Evaluation Example	111
6.4.1	Job and Data Centre Characteristics	111
6.4.2	Quality Evaluation	112
6.5	Conclusions	114

7	Integrated Model	115
7.1	Introduction	116
7.2	Model Integration	117
7.2.1	Hierarchical Infrastructure	119
7.2.2	Thermodynamical model	119
7.2.3	Power and Performance Models	121
7.2.4	Advanced Cooling Control	122
7.2.5	Advanced Power Management	123
7.2.6	Integration in DACSIM	124
7.3	Model Parameters and Output	125
7.3.1	Job and Data Centre Characteristics	125
7.3.2	Simulation Settings	126
7.4	Case studies	126
7.5	Results	127
7.5.1	Energy	127
7.5.2	Performance	129
7.5.3	Thermodynamics	130
7.6	Conclusions	130
II	Experimental Validation	131
8	Workload Modelling for Model Calibration	133
8.1	Introduction	134
8.2	Data Science Approach	134
8.3	Data Preparation	135
8.3.1	Tool Chain	136
8.3.2	Raw Data	137
8.3.3	Processing and Cleaning Data	137
8.4	High-Level Design of Algorithm	138
8.4.1	First Phase	138
8.4.2	Second Phase	141
8.5	Experimental Validation	143
8.5.1	Generated Traces	143
8.5.2	Real Data Centre Traces	147
8.5.3	Comparison with Calibrated Models	149
8.6	Conclusions	150

9	Cross-Model Validation	151
9.1	Introduction	152
9.2	System Description	153
9.2.1	Case Study: A Data Centre Performance Cluster	154
9.2.2	Policy Effectiveness	155
9.3	Power and Performance Model Assumptions	155
9.3.1	Characteristics for Power and Performance	156
9.3.2	Specification of Job Dispatching Policies	158
9.4	Two Implementations	163
9.4.1	The DACSIM Implementation	163
9.4.2	The iDSL Implementation	165
9.5	Experimental Results	165
9.5.1	Power-Performance Trade-off	165
9.5.2	Validity of the Outcomes	168
9.6	Conclusions	169
10	Experimental Micro Data Centres	171
10.1	Introduction	172
10.2	A Key Application for Data Centres	173
10.3	System Description	174
10.3.1	RASPBERRY PI 2	174
10.3.2	Experimental Setup	175
10.3.3	Software Setup	175
10.3.4	Cluster in Data Centre Server Racks	177
10.4	Cluster Benchmarking	178
10.4.1	Storage and Memory Performance	178
10.4.2	Energy Consumption	179
10.4.3	Network Performance	180
10.4.4	Temperatures	181
10.5	Application-Specific Benchmarks	182
10.5.1	The TERASORT benchmark	182
10.5.2	The PI benchmark	185
10.6	A Small Case Study for Model Validation	186
10.6.1	Case Study: RSA Factoring	186
10.6.2	Experimental Setup	187
10.6.3	Simulated Micro Data Centre	188
10.6.4	Model to Measurements Comparison	189
10.7	Conclusions	189

11 Conclusions	191
11.1 Summary	192
11.2 Revisiting Research Questions	193
11.3 Recommendations for Future Research	197
11.3.1 Modelling	197
11.3.2 Energy Saving Techniques	198
11.3.3 Validation	199
Bibliography	201
List of Publications	225
Index	227
About the author	227

Introduction

1.1 Motivation

In 2006, Al Gore, former vice-president of the United States, created a global awareness and willingness to reduce greenhouse gases by releasing his documentary “An Inconvenient Truth” [135]. According to organisations like NASA [131] these green house gases are the cause for worrying climate change with effects including intensified hurricanes and rapid defrost of the Artic Ocean. The urgency in society to act for the preservation of the world’s ecosystem and to improve quality of life for generations to come has been a strong motivation to many. As roughly one third of these greenhouse gases are caused by the generation of electricity [175], opportunities to potentially save electrical energy significantly contribute to the reduction of greenhouse gases.

In 2012, Neelie Kroes [100], former vice-president of the European Commission responsible for the digital agenda, stated that ICT consumed 8% to 10% of all European electricity, which is approximately the total power consumption of the whole of the Netherlands. Such insights in energy consumption and the consequences thereof has led in society to a stronger focus on green ICT solutions in which saving electricity has becoming increasingly important.

Among these ICT solutions, data centres are nowadays considered to be those facilities with a large number of computers. These data centres are playing a pivotal role in all cloud-based services (e-commerce, social networks, financial services) but also in an ever-growing share of industrial and societal processes, think of smart industry and developments toward e-government. The performance of data centres is crucial for the appreciation of all these services by end-users. At the same time, data centres are responsible for an enormous energy consumption, hence, it is also important to carefully design data centres while taking a multitude of considerations into account. The part data centres contribute to this ICT energy consumption is considerably large, as approximately 1.1%-1.5% of the worldwide electricity was consumed by data centres in 2010 according to [97]; this means that data centres are responsible for approximately 0.4%-0.5% of the worldwide greenhouse gasses. The same study states

Year	'00	'01	'02	'03	'04	'05	'06	'07	'08	'09	'10	'11	'12	'13	'14
Energy consumption (TWh)															
Global	–	–	–	–	–	–	–	–	–	–	219	271	332	350	–
US	29	–	38	–	48	–	62	–	68	–	67	–	68	–	70
EU	18	–	–	–	–	41	–	56	–	–	72	–	–	–	–
NL	–	–	.36	–	–	–	.63	–	.95	–	1.29	–	1.58	–	1.62

Table 1.1: Estimated total annual energy consumption by data centres over the last years for different regions in the world (based on the studies and reports [4], [10], [45], [159], [166]).

that the worldwide power consumption between 2005 to 2010 has increased from 25 GW to 40 GW; an amount that is equal to the energy production of about 40 average-sized nuclear power plants (assuming 1 GW power production per power plant). Table 1.1 shows the estimated magnitude and significant increase of the annual energy consumption by data centres and the size of the contribution by different regions globally, in the United States (US), in the European Union (EU) and in the Netherlands (NL).

Whereas the world's ecosystem is a long term motivation for society to save energy, another motivation that drives changes in industry is the opportunity to lower their total cost of ownership by saving on their energy bills. From scientific perspective, the field of data centres generates many new challenges for investigation using existing theoretical concepts.

Data centres are, in fact, an excellent example of and contribute to the research field of *Cyber-Physical Systems* (CPSs). A CPS is a system in which there is close connection between the physical world through sensors and/or actuators with the cyber world that typically consists of some processing on which control algorithms are executed. The complex system of a data centre forms an interesting mix of highly non-linear dynamics (e.g., the thermal process) and typical cyber components (e.g., load scheduling, processors, sensors and communication network).

Data science has become a trending interdisciplinary field to obtain meaningful insights from data. Such trends are reinforced by the large number of sensors in most ICT systems that are already collecting data. Similarly, data centres collect large amounts of information through sensors for energy, performance and thermodynamical related state information variables that have the good potential to be useful for design decisions.

In the last years, strong effort has been made to invest in techniques to re-

Figure 1.1: Trend of total annual energy consumption by data centres in the United States between the years 2000 and 2014 [159].

duce the worldwide energy consumption. From 2000 to 2006 the annual energy consumption of United States data centres increased from 28.5 TWh to 61.8 TWh [159] (cf. Figure 1.1). Whereas the same study shows that in the years from 2006 to 2014, the annual energy consumption only increased from 61.8 TWh to 69.8 TWh. Whilst the performance demands still increase, the energy consumption seems to stagnate over the years. This relatively small growth in energy consumption comes from efforts among data centre owners to push back the energy consumption of their data centres. According to [159], the three main energy-efficiency improvements that contribute to this flattening are (i) *advanced cooling strategies*, (ii) *power proportionality*, and (iii) *server consolidation*.

The advanced cooling strategies focus on techniques that increase the thermal efficiency of the data centre like hot aisle isolation, economizers and liquid cooling. Power proportionality is achieved with power management software and hardware, whereas server consolidation focuses on running current workload on as few servers as possible, in order to decrease the amount of hardware necessary in the data centre.

Power proportionality, i.e., power consumption is proportional to utilisation, has proven to be one of the three main areas of improvement on data centre energy-efficiency in the last years [33], [159]. Since power management software and hardware has been improving, scaling back idle servers has be-

come a reasonable practice nowadays. Power management involves many strategic decisions based on state information at hand. Such strategic decisions could lead to energy savings, while maintaining a good performance.

1.2 Power and Performance Model-Based Approach

Power and performance models are crucial in assisting the design and optimisation of data centres. The design of such complex systems as the data centre can be very challenging. The ability to model and analyse (i) provides more in depth understanding of the system dynamics, (ii) allows for comparison studies and (iii) provides an estimation of power and performance metrics.

Performance modelling of computer networks has already been extensively studied, e.g., in the books [25], [76], [123], [124]. With these foundations in performance modelling and analysis also data centres can be studied. We distinguish two type of approaches, namely: Simulation and numerical analysis. Simulation analysis for data centres can be done with a wide variety of tools such as CLOUDNETSIM++ [120], OpenDC [90], BIGHOUSE [122], DCSIM [173], CLOUDSIM [31], GDCSIM [72], GREENCLOUD [178] and MDCSIM [112] that often use network simulators such as OMNET++ [140], NS-2 [171] and NS-3 [172]. Numerical analysis has been conducted that use linear approaches [15], [113], [192], Markov chains [64], [65], [69], [102] and stochastic Petri nets [30], [92] to study energy-efficiency of data centres from different perspectives with various assumptions.

Also in the case of power modelling, many models have been proposed and analysed. A recent survey [46] shows an in-depth literature study with more than 200 models of data centre power modelling and power estimation. This survey states that one of the key challenges for future research is (i) a lack of modelling efforts made targeting power consumption of the entire data centre, (ii) many power models are based on a few CPU and server metrics, and (iii) the effectiveness and accuracy of the models are still open questions. Also, a meaningful analysis of a combination of these power and performance models is still open to research.

Especially for the application of energy saving techniques, optimisation problems are solved for both power and performance that could lead to so-called trade-offs. For instance, should we put a server to sleep and save power or should we keep it on such that it can work on (soon) upcoming high priority tasks? In this example, situations exists where power can be traded for performance. If this would only be modelled for an optimal energy consump-

tion, the performance could become really bad and vice versa. So, being aware of both power and performance is crucial for decision making.

1.3 Goal & Research Questions

This research addresses the problem of power and performance modelling and analysis challenges that arise from both power management and advanced cooling in the context of a data centre. These challenges assist in making design decisions that aim to lower energy consumption while maintaining good performance. This involves strategic decisions with regard to energy saving techniques. Therefore, the goal to which this thesis contributes is formulates, as follows:

Goal
Acquire meaningful insights in both the power and performance of data centres to assist energy saving techniques using model-based analysis.

In order to achieve this goal, five research questions are addressed below.

Research Question 1: *How can analysis of numerical and simulation models assist in obtaining useful insight in saving energy in data centres while maintaining good performance?* Of course, each system can be modelled in numerous ways. Therefore, finding a meaningful set of models for the analysis of both energy and performance is core to this thesis. Lessons learned from modelling the data centre at two different levels of abstraction is very beneficial for exploring its current and also potential use in practice.

Research Question 2: *How can power management strategies be structurally specified and evaluated for the purpose of saving energy in data centres with performance constraints using models obtained in the context of RQ.1?* By taking one step forward from solely insight in power and performance in the data centre at different levels of abstraction, additional insight in energy saving techniques is crucial in the exploration of alternative data centre designs. As power management is a top contributor to energy savings in data centres, still many strategic decisions can be made based on the state of the data centre.

Research Question 3: *How can power management strategies be combined in the wider context of thermal-aware models for the purpose of saving even more energy in*

data centres using the models obtained in the context of RQ.1? Now even more energy saving techniques exists, such as advanced cooling to strategically adjust air supply temperatures and job distributions. However, combining energy saving techniques is a more complex situation in which multiple models strive for saving energy at distinct levels.

Research Question 4: *How to calibrate and validate the set of models obtained in the context of RQ.1?* Insights truly become meaningful if the models are a truthful representation of the data centre. Therefore, the models need to be calibrated with validated assumptions and values that could be representative to a data centre. Knowledge of which values are required to be able to apply the proposed models is necessary. The level of validation depends on the level of understanding that is desired from the models, i.e., the more detailed the insights should become, the more thorough the validation needs to be.

Research Question 5: *What can be the role of a micro data centre setup in providing useful insights in data centres? How do these compare to the model-based approaches?* The experimental costs of a micro data centre are much lower than interfering with an actual data centre. These micro data centres can be used to perform small scale tests, discover dynamics for specific applications and provide insight in the model assumptions. Furthermore, it assists in sketching the practical context and potential application of the models.

1.4 Approaches

Recall from Section 1.2 that for the analysis of power and performance, the data centre can be modelled in various ways. This thesis covers two major modelling approaches and introduces an experimental micro data centre. These three differentiate from each other in five areas: (i) *assumptions*, (ii) *accuracy*, (iii) *experiment costs*, (iv) *computation speed*, (v) *experiment safety*. Figure 1.2 and Table 1.2 show the overall approach with an indication of how well the approaches are suited for finding meaningful insights for making data centre design decisions.

Through *real world experiments* the right design of the data centre can be found. All experiments can be performed with the highest accuracy of representing the final state and no assumptions need to be made. However, the consequence of real world experiments are the high cost of expensive (new) equip-

Approach	Assumptions	Accuracy	Experiment Costs	Computation Speed	Environment Safety
Real world	++	++	--	--	--
Laboratory	+	+	+	--	+
Simulation	-	+/-	++	-	++
Numerical	--	+/-	++	++	++

Table 1.2: Overview of all advantages and disadvantages of each approach.

Figure 1.2: The overall approach of acquiring meaningful insights in data centres.

ment, down-time of high-availability services and even chances of accidentally disrupting other crucial processes due to human error. Beside that, installing and running experiments cost a lot of effort and might take long before actual meaningful results are presented. Therefore, real world experiments should be seen as the last stage in making design decisions.

An approach that has very low costs and can be performed safely outside the data centre are the use of models that are suitable for *numerical solutions*. Such an approach allows to make a first rough estimation. These solutions can often be computed rapidly and could also be useful for on-the-fly decisions,

e.g., time critical decisions. An advantage is that the outcomes are often accurate with respect to the assumptions. However, these numerical solutions are often limited by their assumptions leaving these useful for only a small subset of data centre configurations. On the other hand, the *simulation* approach requires fewer assumptions and can also be performed safely outside the data centre. The downside of this approach is the lengthier computation compared to the numerical solution.

Since model validation and insights in actual operation within a data centre is often still too expensive and operation should *not* be interrupted due to high availability services, a *laboratory* setup that represents the data centre at micro-level could be very helpful for making design decisions as well. As this environment is a representation of the data centre it has very few limitations, good accuracy and can have low cost, while still research can be conducted safely outside the data centre. Note that the micro-level causes the system to be different from the real data centre, which could lead to several assumptions and less accuracy. The configuration of such a laboratory environment can still consume a considerable amount of time and meaningful insights can still be very hard to obtain.

The basic idea of modelling both energy and performance is illustrated in Figure 1.3. A data centre serves a stream of jobs from the outside world. These jobs are first buffered. Subsequently, the jobs are scheduled and executed on the basis of job requirements and the available system-internal energy, performance and thermodynamical state information variables. This could involve information on queue lengths, server utilisations, networking bottlenecks, temperatures and humidity, to mention a few examples.

Figure 1.3: The basic model for energy/performance trade-offs in data centres.

1.5 Thesis Structure

1.5.1 Thesis Outline

Figure 1.4 shows the position of the approaches based on their initial abstraction level. From these initial abstraction levels we try to move as close to reality as possible, as long as it provides the insights and modelling/measuring complexity desired.

When more *details* are required or *assumptions* are too strict to obtain desired insights one could consider using another approach that is closer to reality. The

Figure 1.4: The storyline of this thesis.

Figure 1.5: The structure of the chapters.

main reasons to use approaches with high abstraction levels are the often lower experiment duration, complexity and costs.

Figure 1.5 shows the structure of the thesis that follows the storyline at different levels of abstraction subdivided in two parts. Before these two parts, Chapter 2 discusses the necessary background required to better understand these parts. In the last Chapter 11, conclusions are drawn from these parts with the research questions in mind from this introductory chapter.

- The *Modelling* part covers Chapter 3–7. These chapters elaborate numerical and simulations models for the analysis of power and performance in the data centres. This includes modelling infrastructure, servers, and two energy saving techniques.
- The *Experimental Validation* part covers Chapter 8–10. These chapters function as a bridge between the models and reality with steps taken to calibrate and validate the proposed models. Moreover, a low-cost experimental laboratory setup is proposed to represent the data centre fundamentals on a small-scale and its potential for power-performance analysis and benchmarks.

1.5.2 Contributions

This thesis contributes to the area of model-based analysis of data centres in a several ways and are detailed below.

Power-performance models Simple single-server and multi-server stochastic Petri net models are proposed for the numerical analysis of power and performance. A mixture of models is introduced to analyse the power and performance of data centres with discrete-event simulation. These models allow for the analysis of power-performance trade-offs.

Power management strategy evaluation method A structural way to describe power management strategies is proposed based on the existing literature. Several quality aspects are defined and analysed with the aid of the simulation models and the strategy description method.

Combined energy saving techniques A combination of the two energy saving techniques power management and advanced cooling is analysed in the simulation framework.

Model calibration and validation The models are calibrated as much as possible with the workloads available. The power-performance models are

validated with a general-purpose simulator and comparisons are made between the models at different abstraction levels.

Experimental micro data centre An experimental laboratory setup consisting of small servers has been proposed for comparison purposes and power-performance analysis.

1.5.3 Contents and Origin of the Chapters

A brief overview of the chapters including publications from which the chapters originate, is given below. Moreover, this research has been featured in an issue of a well-known Dutch magazine [B8].

Chapter 2 presents the background on *data centres* including an elaboration of power management techniques. Furthermore, modelling and analysis approaches are elaborated that form a basis for the following chapters.

Chapter 3 presents the *numerical solutions* for the analysis of both power and performance in data centres with power management features. Moreover, the power-performance trade-off is discussed and several estimates are provided for a variety of data centre configurations. The work is based on the publication [B10].

Chapter 4 presents a *simulation* framework with power and performance models for the analysis of power and performance in data centres with power management features. Several data centre configurations are simulated to illustrate the versatility for the analysis of both power and performance. The work is based on [B7].

Chapter 5 presents a *specification for power management strategies* based policies described in the literature. These literature-inspired approaches are simulated for both power and performance. This work is based on [B4].

Chapter 6 presents an *evaluation method for finding power management strategies* with good quality. This work is based on [B2].

Chapter 7 presents a *combination of saving energy with power management and advanced cooling techniques*. This leads to an integrated model that is simulated with four scenarios. This work is based on [B3] that has been performed in collaboration with research project partners from RIJKSUNIVERSITEIT GRONINGEN.

Chapter 8 presents a way to *calibrate* the models using a data science approach on realistic workload traces. This work is based on [B1] that has been performed in collaboration with BETTERBE and a supervised student.

Chapter 9 presents steps for *validation* of the models including a comparison of the simulation framework to a general purpose model checker and a comparison between the numerical and simulation models. This work is based on [B5], which also served as basis for a chapter in PhD thesis of Freek van den Berg [20], that focusses on the capabilities of the general model checker.

Chapter 10 presents an experimental *micro data centre* for the analysis of power and performance. This work is based on the publication [B6], and served in bachelor publications from five students supervised by us.

Chapter 11 presents the *conclusions* by revisiting the research questions and discussing the most relevant contributions.

Data Centres

This chapter adds background on data centres and important energy saving techniques, for a better understanding of the upcoming chapters. For the data centre, historical background is provided together with definition of what is considered a data centre. Since large amounts of energy are consumed by data centres, the study is extended with discussion on the three major energy saving techniques of the last years. An overview is provided of modelling and analysis approaches, since it has been a proven way to assist in developing these energy saving techniques.

This chapter is organised, as follows. First, data centres are put in historical perspective and provided with a definition in Section 2.1. In Section 2.2, the data centre infrastructure is elaborated. Section 2.3 elaborates the energy saving techniques. A modelling and analysis approach overview is provided in Section 2.5.

2.1 The Rise of Data Centres

At some point in time “data centre” became the two words to describe the facilities that contain large number of computers. The essential historical events up to the point the so-called data centres emerged is elaborated in Section 2.1.1. After description of the origin of data centres, a more exact definition of what we will consider to be a data centre is provided in Section 2.1.2. Section 2.1.3 then addresses proven energy saving techniques in data centres and highlights the current main contributors to saving energy in data centres.

2.1.1 History of Data Centres

Data has been collected and examined for many years by organisations. Starting as early as the *punch cards*, in 1890, which allowed summarising data with the Herman Hollerith tabulating machine. Later on between 1900-1950, similar machines would be offered as a service and even terms as “Super Computing Machines” were already introduced in [43], [54]. In 1947, the United States Army patented one of the first *supercomputers* called Electronical Numerical Integrator and Computer (ENIAC) [148]. The UNIVersal Automatic Computer I (UNIVAC I) was introduced in 1951 in the United States [136] that would later become to what now is called a *mainframe* computer. While these supercomputers were designed for fast computation of large tasks with a strong focus on high performance, mainframes on the other hand were designed for many concurrent smaller tasks and often required high availability.

The *Internet* grew in popularity starting from 1969 with the introduction of the ARPANET in the United States for military purposes [109]. As networks of computers started to grow worldwide, at some point the *client/server architecture* was introduced to share workload among different resources. Such developments eventually lead to so-called *minicomputers* that would use other “bigger” computers with larger processing and/or storage capacity. In 1981, the consumer version of the computer called the Personal Computer (PC) boosted the microcomputers industry [138]. Networks of such microcomputers would later on replace the mainframe computers in the 1990s, which are now the so-called *servers*. Since then, servers have formed the backbone of data centres.

2.1.2 Data Centre Definition

After a discussion about the origin of data centres, a definition is provided (inspired by [163]) below.

Figure 2.1: An example data centre with a room full of servers stacked up in racks (a) and closely connected by some network equipment (b).

Definition 2.1. A *data centre* is a physical location that facilitates a group of networked computer servers used by third parties to store, process and/or distribute data.

While there are more comprehensive definitions available [9], [17], Definition 2.1 is complete, short and resembles a well-thought-out definition from [189, p. 153–155], which defines a data centre, as follows:

The data center is a place where [*sic, recte* which] can accommodate many computing resources that collect, store, share, manage, and distribute a large volume of data. It consists of all necessary data center facility elements (space, power, and cooling) and IT infrastructure elements (server, storage, and network) based on business requirements.

A data centre often facilitates a number of servers and network equipment stacked up together in racks. Such IT equipment requires many infrastructural components to be able to operate and preserve the right operating conditions.

A typical data centre houses thousands to tens of thousands of servers within a large hall with additional infrastructural components to be able to operate and preserve the right operating conditions. Currently, there are already numerous of such data centres worldwide.

2.1.3 Saving Energy in Data Centres

As a consequence of the high demand for a growing number of applications, many new data centres did appear recently to house all these servers. These

additional data centres entail a large energy consumption. Consequently, great benefits could be obtained with energy saving techniques.

A number of ways to decrease energy consumption in data centres are being researched and applied according to [56]. These ways range from very practical (“moving boxes”) to more advanced (using elaborate sensory equipment) and more software-oriented approaches. An overview of their means to decrease energy consumption is provided below.

1. Data centre ICT equipment:

- (a) The use of *low voltage processors* will directly decrease the power usage by some 30%, thereby not necessarily impacting performance. These type of processors have already become a common practice with well-known retailers [49].
- (b) The use of *high-efficiency power supplies*, with efficiency 90% instead of the typical 70% that is standard for low-end power supplies in consumer computing equipment. Furthermore, the power supplies should be chosen such that they, under normal load circumstances, operate at their optimum. Even higher power efficiencies of up to 92% are achieved by changing the conventional architecture of the data centre infrastructure [103].
- (c) The use of *blade servers*, that minimises energy consumption and physical space with a modular and more efficient design compared to standard rack servers. The study [77] confirms this statement, as an energy reduction of 5.2% is expected for the blade servers compared to standard rack servers. However, a rack full of these high-density blade servers consumes 33% more energy and has significantly higher performance than a rack full of rack servers.
- (d) The use of emerging techniques for *green networking*; in most of the data centre literature there appears to be a strong focus on computing only. The techniques for green networking that reduce energy are more thoroughly discussed in the survey [23], and roughly categorised, as follows: (i) adaptive link rate; (ii) interface proxying; (iii) energy-aware infrastructure; and (iv) energy-aware applications.

2. Data centre (power) management software:

- (a) The use of *advanced power management software*, that make that servers or server groups can be switched off completely while still meeting the performance requirements (more on this in Section 2.3.2).
- (b) Advanced *server virtualisation* software can be used to increase server utilisation and to reduce the number of active servers (more on this in Section 2.3.3).

3. Data centre power supply:

- (a) Higher voltage AC *power distribution* within a data centre can decrease overall power usage, as higher voltage transport is more efficient than low voltage transport; the EU is doing better here with their standard 240 V than the US with 110 V according to [151].
- (b) The use of *more efficient UPS systems*, that do avoid the double conversion, from the external AC source, to the DC storage and buffering, and the AC end-use. According to [186], multiple high-density data centres have already moved their UPS systems closer to the servers to reduce energy losses due to conversion.

4. Data centre cooling:

- (a) The use of better *spatial arrangement of servers* and cooling (use of hot/cold aisles) and higher room temperatures (28 vs. 20 degrees Celsius) and *variable capacity precision cooling* (instead of simple overall room cooling) to cool just there where it is needed (more on this in Section 2.3.1).
- (b) *Per server/system monitoring and control* of temperature, humidity, etc., to further increase cooling efficiency; this requires the installation of an advanced (wireless) sensor system (more on this in Section 2.3.1).

According to [159], the three main energy-efficiency improvements that contributed to a flattening in the energy consumption in the past few years are (i) *advanced cooling strategies* (4.b. in the list above), (ii) *power proportionality* (2.a. in the list above), and (iii) *server consolidation* (2.b. in the list above). This thesis focusses on these three, with most emphasis on power proportionality, i.e., servers consume power in proportion to the amount of useful work performed, a goal that has first been introduced in [13]. In general, power proportionality is achieved by a combination of *power management* and *load balancing* techniques.

Figure 2.2: Overview of typical data centre infrastructure components.

2.2 Data Centre Infrastructure

In this section, the data centre infrastructure is discussed to show useful details in the context of modelling power and performance. This context is also necessary to understand important data centre design decisions. First, Section 2.2.1 describe typical components one encounters in a data centre. Most of the components are the result of various data centre demands driven by certain application domains, which are discussed in Section 2.2.2.

2.2.1 Components

Each server in a data centre has its own set of demands. The minimum requirements for a server is location, space, power supply, network accessibility and healthy environment conditions. In order to ensure location and space, servers are stacked up in racks. Furthermore, the right amount of power needs to be distributed to each server in all racks.

To guarantee network accessibility, servers are either interconnected, connected to the Internet or connected to some other network. Servers with healthy environment conditions require certain temperature and humidity to prevent unnecessary server depreciation. This is achieved by installing cooling and chillers components.

Figure 2.2, which is inspired by [14] and [143], shows the various components that are typically used in a data centre to operate under these requirement.

Power is either delivered from the grid or the power generator depending on state of the *Automatic Transfer Switch* (ATS), which enables to switch between the two power sources, i.e. if one power source fails the other power source instantaneously takes over. Next, the power flows through *switch gear* to isolate, protect and control it. The power is transferred to the data centre *Uninterruptible Power Supplies* (UPSs). These UPSs can deliver enough power to supply the data centre for a duration that is long enough to activate and stabilise the power supply from the *power generator*. UPSs are often backed up with extra *batteries and/or a flywheel* in the case that one UPS fails. The UPSs are connected to *Power Distribution Units* (PDUs), which supply power for all the *servers, chillers, cooling, networking, monitoring and control components*.

2.2.2 Important Data Centre Demands

In [9], important demands for data centres are distinguished, that direct choices on architecture, namely: *availability, scalability, flexibility, security and performance*. A short description of each demand is given below.

Availability Ensure no single point of failure exists and uptime is predictable.

Scalability Ensure support for fast growth without large-scale intervention.

Flexibility Ensure support for new services without huge adaptation of the infrastructure.

Security Ensure attack prevention by firewalls, authorised access and protection of data.

Performance Ensure optimal performance by servers, topology and load balancing.

Data centre owners often explicitly state the service delivered to their customers in *Service-Level Agreements* (SLAs), which describe a contract between an owner and a customer. Such SLAs describe performance indicators and quality demands for a service or product. Besides some description of services and responsibilities, a typical SLAs in data centres contain agreements on: (i) availability, (ii) temperature, (iii) humidity, (iv) response times, (v) traffic quota, (vi) disk quota, (vii) bandwidth, (viii) support, (ix) maintenance, and (x) penalties/consequences for not meeting the demands.

Often data centres need to comply with well-known standards, e.g. *ISO* [91], *NEN* [133] and *EU Code of Conduct for Data Centers* [58]. Data centres that

claim to meet the requirements of the standard then have to undergo inspection from independent third parties.

Example 2.2 (SLA demands for three application areas). The SLAs vary between application areas. Table 2.1 shows the importance of specific demands to three example application domains.

	Large research computation problems	Hospital information systems	Office data back-up
Availability	--	++	--
Temperature	++	+/-	+/-
Humidity	++	+/-	+/-
Response times	++	+/-	+/-
Traffic quota	++	+/-	++
Disk quota	++	+/-	++
Bandwidth	++	+/-	++
Utilisation	++	+/-	--

Table 2.1: Three example application areas with potentially very different data centre requirements (low priority = --, medium priority = +/-, high priority = ++).

High performance computing is often characterised by powerful hardware for complex and/or parallel computations (e.g., research with large computation problems). These high performance computing clusters are often used to solve large computation problems in scientific research.

Hospital information systems often require applications with critical tasks that require high availability (e.g., the retrieval of patient laboratory blood tests and medical background in emergency situations). These applications often require to be operational at all times, because a good communication in a hospital is crucial when human lives are at stake. While for thermal conditions and performance of the data centre, only medium priority suffices.

Many businesses have the need for office back-up utilities that requires data centres with much storage capacity to store large amounts of data. For such businesses, bandwidth, traffic quota, disk quota should be large. However, back-ups can be precisely scheduled such that utilisation and availability is of less concern as long as data integrity is preserved.

This example illustrates the variety of requirements that can exist between different application domains that could result in different infrastructures. □

2.3 Power Saving Techniques

Recall from Section 2.1.3 that the three main energy-efficiency improvements are (i) *advanced cooling strategies*; (ii) *power proportionality*; and (iii) *server consolidation* according to [159]. Section 2.3.1 elaborates advanced cooling. A way to achieve a better power proportionality is elaborated in Section 2.3.2. The third energy saving technique, advanced server consolidation through virtualisation, is discussed in Section 2.3.3.

2.3.1 Advanced Cooling

In a data centre, much heat is produced by the UPSs, PDUs, lights, people and IT equipment. According to [189], IT equipment is responsible for about 73% of the heat production. The main functionality of cooling is to prolong the lifetime and to maintain operation of the IT equipment in the data centre. Many traditional data centres base their environmental operating conditions on the ASHRAE guidelines [5] to determine acceptable temperature and humidity for their heat producing equipment.

The main role of a *cooling* (CL) system is to maintain the right temperature and humidity in the data centre by extracting heat from the IT equipment by transportation of cold air through it. A data centre can be equipped with components to maintain the right temperature and humidity like (i) computer room air conditioning (CRAC) units; (ii) computer room air handling (CRAH) units; (iii) chillers; and (iv) humidifiers [189]. Temperature and humidity can be monitored and maintained in a data centre via mechanical refrigeration to cool air with a CRAC unit. Traditionally, the humidifiers are incorporated in these CRAC units. Alternatively, the CRAH unit has fans that blow air over a cooling coil filled with chilled water to cool the data centre.

These traditional *active* CL methods consume a large amount of energy, because (i) these CL systems have continuous operation, day and night, all days of the year; (ii) energy is consumed by the pipe system by pumps and fans to transport water and air; (iii) mixing of cold and hot air flows leading to inefficiencies [191]. The mixture of cold and hot air flows can be improved by creating hot and cold aisles and by providing proper airflow control devices.

Moreover, many *passive* CL methods exist to provide a source of "free" CL in certain occasions by reducing energy consumption using outside air, water and heat pipes [191]. These methods depend heavily on the weather conditions at the geographical location of the data centre and are consequently only applicable under the right conditions.

Internally, more advanced CL techniques could be applied by correctly distributing tasks to colder systems and thereby reducing the energy consumption [129], [164], [187]. In this thesis, advanced CL techniques are also taken into consideration in Chapter 7.

2.3.2 Advanced Power Management

In this subsection, we consider *Power Management* (PM) involving the efficient direction of power to the server components. First, the various PM techniques are elaborated. More details on how power state switching allows these PM techniques to function inside a server are then elaborated.

Power Management Techniques

The PM techniques describe the capabilities that are within a server to efficiently direct power to server components.

PM takes place at various abstraction levels within a server. The lowest level of PM, i.e., PM closest to the hardware, discussed here is concerned with individual devices within a server. An example of this, concerning processors of a server, is *Dynamic Frequency and Voltage Scaling* (DFVS). By decreasing the frequency and voltage, when a server is idling, it reduces the power consumption while keeping the performance intact. Governors control the clock frequency and voltage with smart strategies to make sure the performance meets its requirements, while saving a lot of energy.

A higher level form of PM, is on the level of each individual server. An entire server can be switched to a lower power state to decrease the energy use by suspension of devices making them unavailable, and thereby decreasing system functionality. A potential reason for doing so, is a server that is being underutilised.

The survey [126] on PM techniques for data centres distinguishes and studies four PM techniques: (i) DFVS; (ii) transition to low-power states and server consolidation; (iii) workload management or task scheduling techniques; (iv) thermal-management or thermal-aware techniques including cooling related issues of data centres. These studies focus on the reduction of the overall energy, reduction of peak energy consumption and power capping (limiting the power consumed) in some cases under certain performance constraints. Also, several model-based and control-theoretical approaches are mentioned.

The key question with PM techniques is often: What is the right moment to reduce performance capacity and save energy? The answer to such a ques-

tion relies heavily on the context. For instance, jobs in a data centre might be distributed in such a way, that the load on a server is decreased for a short period. It would be beneficial to reduce processor's capacity to save some energy. However, if the load on a server is decreased for a longer period, it might be beneficial to put servers in a sleep state.

Hardware suppliers developed solutions that automatically perform PM on a large number of servers. Currently, monitoring and control is very common via the combination of software for remote management and specialised servers with useful sensors and/or other intelligent equipment. Server provisioning is often done via scripts and/or virtualisation. The current main focus of most companies lies with easily adding, migrating and removing servers and/or virtual machines.

Power State Switching

This section elaborates on the dynamics inside a server that allow for the PM techniques to be able to operate. A server switches internally between so-called *power states* to enable or increase system functionality, that again consume an additional amount of power.

Low power states are used to reduce the energy consumption of individual servers, while (partial) functionality of the system is decreased. Most operating systems support the open standard specification *Advanced Configuration and Power Interface* (ACPI) for device configuration and PM, which is meant to replace the older PM Application Programming Interface (API) called *Advanced Power Management* (APM) [89]. Compared to APM, ACPI shifts the power management responsibilities from the *Basic Input/Output System* (BIOS) to the *operating system* (OS), because the OS has more knowledge of the application and system compared to the BIOS.

A global overview of ACPI is depicted in Figure 2.3 inspired by [79]. The *Operating System Power Management* (OSPM) enables to implement the most efficient power mode, which makes use of the ACPI interfaces to switch between power, performance and processor states. The figure shows how software and hardware components relate to each other's interfaces. *ACPI Tables* are the central data structure of the ACPI-based system, which describe all hardware that can be managed by ACPI. *ACPI BIOS* performs low-level management operations on hardware. These operations help to boot, sleep and wake the system. *ACPI Registers* are hardware management registers based on the ACPI specification and addresses are kept in the ACPI tables.

In more detail, ACPI specification has four *global states* (Gx) and five *sleep*

Figure 2.3: Overview of ACPIs connection to hardware and the operating system.

Figure 2.4: Switching between global (Gx), sleep (Sx), device (Dx), processor (Cx) and performance (Px) states in the ACPI specification.

states (Sx), which hold for an entire server. The cooperation between the global and sleep states is depicted in Figure 2.4 inspired by [79]. The legacy state disables ACPI, such that the power and hardware is not managed via ACPI.

The first global state (G0) is a sleep state as well (S0), and represents the *working* state, where the monitor is off, but background tasks are still running. The second global state (G1) is subdivided into four sleep state. The first of the sleep states (S1) is the *Power on Suspend*, in which all processor caches are flushed and the CPU stops execution of instructions. Still power is supplied to the CPU, RAM and devices that explicitly state that they require to be on. The second sleep state (S2) powers the CPU off and dirty cache is flushed to RAM. The commonly known *Standby*, *Sleep* or *Suspend to RAM* state is the third sleep state (S3). *Hibernation* or *Suspend to Disk* (S4) is the last of the four states, where all content of the main memory is saved to disk, e.g., a hard drive. Whenever this thesis refers to *advanced power management*, the concept of switching between global and sleep states is used. In this thesis, advanced PM is featured in the models proposed in Chapter 3–4.

Furthermore, ACPI provides to switch between *device state* (Dx), *processor state* (Cx) and *performance states* (Px).

The device states differs from *fully on* (D0) to the two states (D1 and D2), which are defined by each individual device, to *off* (D3). The off state has a hot and a cold alternative, where in contrast to the cold state, the hot state allows the device to switch to higher (D0 to D2) power states.

The processor states are the *operating* C0, *halt* C1, *stop-clock* C2 and *sleep* C3 state. C1 does not allow the processor to execute any instructions, but can resume any time and is activated by software. In contrast with C2, which is activated by hardware, which just as C1 stops the CPU core clock, and thus reduce power consumption. In the previous states the bus interface unit and the *Advanced Programmable Interface Controller* (APIC) are still running, such that the CPU can handle interrupts and an external bus. In the C3 state, the bus interface unit and APIC are cut off from signals from the CPU. Manufacturers are free to add more processor states.

The performance states are implementation-dependant, since these states are meant for scaling the voltage and frequency, where P0 is the maximum power and frequency and P1, P2, ..., Px states scale the voltage/frequency. This is done to reduce power consumption by reducing the voltage/frequency, and to enable overclocking and in rare cases improve reliability by increasing the voltage/frequency. The power saving technique *dynamic voltage and frequency scaling* (DVFS) refers to power management at processor level and typically switches strategically between these performance states.

ACPI in practice is often even more complex. For instance, systems also need to be able to reboot via ACPI. This is solved by a register dedicated to ACPI, which is set to a specific value to indicate that the system should reboot. This register is then monitored, and when it contains the rebooting value it resets all components and boots.

The main goal of power management is to reduce energy consumption, while performance is kept intact. The above techniques are often applied to enable this. However, still strategic decisions need to be made on when to switch between power states. These strategic decisions are often based measurement data obtained from sensors that are often installed for meeting certain data centre demands. We have provided a specification to structurally describe and evaluate these strategic decisions in Chapter 5–6.

2.3.3 Advanced Server Consolidation

Server consolidation (SC) advocates more efficient usage of the number of servers required in a data centre resulting in energy reductions. In [183], server consolidation exists in two variants, namely: (i) static SC, and (ii) dynamic SC. *Static SC* is concerned with the reduction of the number of resources over long periods of time, e.g., predictable demand patterns such as peak utilisations; whereas *dynamic SC* reduces the number of resources based on shorter periods of time, e.g., migration of services based on time-varying and stochastic workloads. Despite static SC already improves efficiency, this still leads to resource overprovisioning most of the time. However, short term decisions can improve the efficiency even more through dynamic SC.

Virtualisation enables to consolidate servers by allowing multiple virtual resources to be executed on a single physical resource, thereby reducing the number of physical resources necessary for operation. In practice, resources can be virtualised at hardware, storage and network level.

Hardware virtualisation allows to create *virtual machines* (VMs) on a single physical machine, where each VM has the functionality of a single computer with an operating system. Moreover, each VM instance comes with some overhead and causes *performance degradation*. Also, (live) *migration* of VMs, that is the process of moving a VM from one physical machine to another, for efficiency purposes has temporal negative impact on system performance.

A survey on virtual machine placement techniques in cloud data centres [180] summarises the main challenges with virtualisation, as follows: (i) performance degradation, (ii) (live) migration, (iii) prediction of server energy consumption, and (iv) minimising energy and meet SLAs, and (v) efficient

server consolidation is considered a computational complex problem. These challenges often easily overlap with the other energy saving techniques, e.g., advanced PM and advanced cooling), and are to be taken into account jointly.

Example 2.3. A company decides to host their web services in a data centre. The company would like to host 15 different small web services and 3 bigger web services. Initially, four servers without any VMs are proposed. A data centre specialist discovers that the three small web services have a very low utilisation. The company receives advice to consolidate the number of servers decreasing the energy consumption and reducing their overall operational costs. Now only two servers are proposed that run four VMs, where one server is dynamically activated based on network traffic observations. $\square$

Dynamical activation of servers, as illustrated in Example 2.3, involves complex strategic decisions. *Advanced SC* through virtualisation determines the right number of VMs on each physical machine, often based on hardware utilisations, network traffic and cooling aspects while taking performance degradation and live migration into consideration. In recent years, many solutions have been proposed in the literature to solve these complex decisions. A study on virtualisation in data centres [183] already indicates over 18 research papers with different (mostly dynamic) SC approaches, where each approach is characterised with specific objectives that takes performance, reliability, certain hardware, network traffic, cooling aspects, and/or migration overhead into consideration.

2.4 Load Balancing

Job dispatching involves the distribution of tasks to the available servers. The way jobs are dispatched is important for good performance, and thus has an indirect impact on the energy consumption. Improving the distribution of tasks, also referred to as *load balancing* (LB), can lead to better performance, thereby reducing energy consumption. Therefore, the various existing and proven LB techniques are elaborated first. Secondly, the challenges of LB in energy- and performance-aware data centres are discussed.

2.4.1 Existing Load Balancing Techniques

Dispatching of jobs takes place at multiple levels upon arrival through the network. The first LB in a data centre takes place in a job dispatcher, that distributes the load to the different racks or servers. Inside these racks and servers

jobs are again distributed. Even at server-level jobs are scheduled to local resources (e.g. process scheduling). For that reason, well-known LB policies have been developed to automate decision-making and deliver high performance. The LB policies discussed below and used in this thesis are similar to the algorithms described in the book [44], that explains the practices of load balancing in data centres at a well-known large company.

The most widely used LB policies are made in such a way that distribution of jobs achieve the highest performance. The *Round-Robin* policy is one of the best known, where jobs are distributed in circular order among all resources. This approach often only works well for server clusters with homogeneous servers. For clusters with heterogeneous servers, the *Weighted Round-Robin* policy offers an alternative, as it assigns higher weights to servers that have more processing capacity. The Round-Robin policy is fair with respect to the distribution of the number of jobs. However, as jobs vary in size this can be undesirable. In those cases, the *Fair Queueing* policy considers the size of the jobs. Also, the *Shortest Queue Next* policy (similar to the *Least-Loaded Round-Robin* policy from [44]) adds some fairness by assigning jobs to the machine with the least number of jobs in the queue waiting to be processed. Of course, the *Random* policy is also among the algorithms alternatives, where a random number generator decides to which resource jobs are allocated.

These well-known policies are also taken into account in this thesis of which later on some are formally described and implemented with a policy language for studying load balancing techniques in data centres in Chapter 9.

2.4.2 Load Balancing in Energy Saving Data Centres

Power proportionality is achieved by a combination of energy saving techniques and LB, e.g., while power management decides on the power supply and operation level of resources, LB occurs parallel to power management and involves the assignment of jobs to the right resources. Often, load is balanced among available resources to obtain good performance. However, for the purpose of saving energy, load could be distributed otherwise to attain better energy-efficiency, e.g., load distributed based on server rack temperature to alleviate the cooling system.

Consequently, finding the right resource(s) involves decisions based on more state information than existing and proven LB techniques currently offer. As a consequence of extending data centres with energy saving techniques, LB involves decisions that include the operational status of the server, thermodynamics and the arrival process in order to meet the various data centre de-

mands, e.g., no jobs should be allocated to suspended servers (unless the delay is acceptable). In Chapter 7, also the thermodynamics are taken into account.

2.5 Modelling and Analysis

Section 2.5.1–2.5.4 starts with highlighting various state-of-the-art modelling and analysis approaches that exist in the literature. Section 2.5.5 provides a structural overview of the various modelling and analysis approaches, whereas in Section 2.5.6 the positioning of this thesis is discussed.

2.5.1 Linear Models

These models describe the system with linear equations. These models are solved numerically using algorithms.

In [15], [16], [53], linear models for the analysis of energy use in virtualised data centres have been proposed as part of the Fit4Green project [95]. A framework has been developed for energy-aware placement of virtual machines in data centres. This framework allows for the computation of the best placement of servers under certain constraints found in the SLAs. The placement is then optimised for lowering the energy consumption, thereby contributing to the field of advanced server consolidation. Moreover, the approach supports heterogeneous multi-core servers that are stacked in a data centre. Workloads are synthetically generated. The framework can potentially have long execution times. However, by splitting the problem and using parallel computation these long execution times can be reduced.

The literature on advanced power management distinguishes two modelling styles: (i) predictive and (ii) reactive-control. These modelling styles determine the right number of servers, i.e., power is made proportional to the amount of workload. *Predictive* models use workload patterns to determine the right number of servers, whereas *reactive-control* models focusses on current observations to set the number of servers right. In [35], [70], [71], [82], [99], [119], [125], [179], models are used for data centre environments to predict load in combination with server provisioning algorithms. In [59], [70], [71], [82], [110], [132], models are used for data centre environments to set the number of servers right with current observations.

Much of the literature focusses on computational servers, several sources [26]–[28], [130] in the literature discuss the challenges of storage systems in data centres and propose models for systems with advanced power manage-

ment features. Storage systems have specific challenges related to recovery after reboots, power failures, and internal volatile memory losses.

2.5.2 Control-Theoretical Models

Control-theoretical models describe a system with differential equations for continuous-time models and with difference equations for discrete-time models. These systems are formulated as optimisation problems and are often solved numerically using control-theoretical algorithms.

For proper control of a system, the system states are modelled with some input and output parameters. Moreover, internal functionality is known to describe future states of the system. These approaches are often used to describe the optimisation problems that involve multiple objectives (e.g., low power consumption and good quality of service) that can be controlled with one of the energy saving techniques.

For dynamic capacity provisioning in data centres, control theoretical approaches are extensively studied that formulate and solve optimisation problems with regard to dynamical capacity provisioning [3]. Algorithms (e.g., Receding Horizon Control) are often applied for various scenarios as a method to predict workloads based on daily patterns [7], [113], [114], [192] to decide on the allocation of available resources (e.g., server pools and queues) of resource consumers (e.g., tasks and packets).

2.5.3 Markov Chains

These are stochastic models with a memoryless property that describe systems with states and transitions, e.g., discrete-time Markov chains. Markov chains are used to model the assignment of servers dynamically in data centres with advanced power management.

In [64]–[67], Markov chain models are numerically analysed for dynamic server provisioning for multi-tier data centres with power management functionalities. The work includes powerful algorithms to save energy while maintaining performance. In [69], power and performance are studied with a Markov reward approach for IaaS cloud computing applications that uses virtualisation techniques. As a results of various configurations for these techniques, interesting power-performance trade-offs are discovered. In [102], a generic queueing model for advanced power management that switches between power states using a *finite state machine* is used.

2.5.4 Petri Nets

These models describe concurrent processes in systems using places, transitions and arcs, e.g., stochastic Petri nets for servers switching between power states. Petri net approaches have been used to analyse data centres with power management and virtualisation techniques.

In [29], [30], energy-aware resource allocation is been determined using *stochastic reward nets*, which is an extension of generalised Petri nets. These Petri nets are used to model the power management and virtualisation functionalities. In [92], generalised stochastic Petri nets and Markov automata are used for power and performance analysis of computational servers for data centres with virtualisation.

The advantage of these Petri net approaches is the intuitive meaning of these models compared to Markov chains. Markov chains tend to become large, while Petri nets remain rather compact. On the other hand, Petri nets might have limiting assumptions with regard to the arrival processes and often depend numerically on Markov chains to be solved. The existing approaches using Petri nets are currently mostly focused on virtualisation.

2.5.5 Modelling and Analysis Approaches Overview

An overview of a collection of modelling and analysis approaches (as most already discussed in more detail in Section 2.5.1–2.5.4) is provided in Table 2.2. This collection is provided to highlight the variety of the existing approaches and a rough categorisation to the best of our knowledge. This collection allows us to position and discuss our approach better w.r.t. existing literature. Each set of publications is characterised by (i) modelling approach applied; (ii) analysis approach applied; (iii) energy saving techniques studied; (iv) environment targeted/supported for evaluation; and (v) arrival process. Our thesis contributes to this and distinguishes itself from the existing literature by proposing solutions that use an unique combination of these characteristics as the last two rows in the table indicate.

2.5.6 Discussion

So far, the literature has already proposed many modelling and analysis approaches to contribute in the area of energy saving techniques in data centres for various configurations and workload characteristics.

Many modelling and analysis styles have been applied with varying advantages and disadvantages. Classical numerical solutions in the field of per-

formance evaluation tend to need stronger assumptions. Note that, these assumptions can be fine as long as the estimates are still accurate. For this type of approaches, validation and sensitivity analysis is currently an important next step. Since analysis of advanced PM that combine both power and performance aspects is still open for research, Chapter 3 contributes to the existing numerical solutions by analysis of both power and performance insights and trade-offs using stochastic Petri nets.

Simulation allows for more flexibility with regard to the assumptions, e.g., a more realistic arrival process, while still considering models with a detailed level of abstraction, e.g., support of multi-core servers. On the other hand, simulation has often lengthier computational times. Chapter 4 describes our sim-

Publication(s)	Modelling Approach	Analysis Approach	Energy Saving Considerations	Targeted Environment	Arrival process
[15], [16], [53], [95]	Linear	Simulation	Advanced SC	Heterogeneous multi-core stacked servers in a data centre	Synthetic
[41]	Linear	Simulation	Advanced CL	Heterogeneous stacked servers in a data centre	Synthetic Realistic
[113], [114], [192]	Control-theoretical	Numerical	Advanced PM	Heterogeneous servers	Realistic
[29], [30]	Petri net	Numerical	Advanced SC Advanced PM	Homogeneous servers	Exponential distribution
[92]	Petri net Automata	Numerical	Advanced SC	Homogeneous servers	Exponential distribution
[63]–[67]	Queueing Markov chains	Numerical	Advanced PM	Multi-tier data centres with various configurations	Any distribution Realistic
[69]	Markov chains	Numerical	Advanced SC	Multi-tier data centres with various configurations	Exponential distributed
[102]	Queueing Markov chains Automaton	Numerical Simulation	Advanced PM	Homogeneous servers	Any distribution
Chapter 3	Queueing Markov chains Petri nets	Numerical	Advanced PM	Homogeneous servers in a data centre	Exponential distribution
Chapter 4–8	Queueing Agent-based Automata	Simulation	Advanced PM Advanced CL	Heterogenous multi-core stacked servers in a data centre	Any distribution Realistic

Table 2.2: An overview of a collection of modelling and analysis approaches.

ulation approach that is presented with an extensible framework for a mixture of agent-based models, queueing models and automata. The featured models are extended with support of a specification and evaluation of advanced PM strategies in Chapter 5–6.

Furthermore, multiple extensive surveys [52], [127] have been conducted that categorise the literature around the major energy saving techniques in data centres and propose directions for future research. Interestingly, already the identification of workload patterns has potential to improve energy-efficiency in data centres. Moreover, both studies suggest that combinations between job scheduling, power management, virtualisation and cooling still hold great potential and could lead to more energy reductions in data centres.

As only few approaches in the existing literature combine energy saving techniques, the simulation framework is later on extended to support simulation of two major energy saving techniques by also considering advanced CL in Chapter 7. Workload patterns from a real data centre are taken into account to calibrate the simulation models in Chapter 8.

Data centres often have a risky environment and high experiment cost, which make it often less accessible for the validation cycle (cf. Table 1.2 and Figure 1.2). Alternatively, in this thesis, we explore the potential of a laboratory environment for the purpose of estimating the behaviour of a real data centre in Chapter 10.

2.6 Conclusions

Data centres facilitate a group of networked computer servers used by third parties to store, process and/or distribute data. With the emergence of data centres, large amounts of energy are being consumed as a result.

Infrastructural design choices have a great impact on the overall power and performance of a data centre. Moreover, many important demands drive choices in the number and type of components necessary in the data centre.

Even with optimal infrastructural design choices, other techniques contribute greatly to saving energy with the three major contributors (i) advanced cooling strategies, (ii) power proportionality through power management and load balancing, and (iii) server consolidation through virtualisation. All these energy saving techniques require advanced strategic decisions that concern optimisation and power-performance trade-offs.

Modelling and analysis has an important role in defining these strategic decisions for data centres with high demands. Many modelling and analysis

techniques exists that are roughly categorised. The upcoming chapters contribute to this research area and addresses various research opportunities with regard to model-based analysis of (i) both power and performance, (ii) power-performance trade-offs, and (iii) combined energy saving techniques. These chapters also extend towards (i) an advanced PM strategy evaluation method, (ii) model calibration and validation if possible, and (iii) an experimental laboratory setup for data centre application dynamics and comparison.

Part I

Modelling

Stochastic Petri Net Models: A Numerical Approach

This chapter contributes to the early design phases of data centres by providing insight into the power-performance trade-off that arises from power management (PM). This chapter proposes a flexible set of stochastic Petri net models which can be used easily to study the trade-off between performance and power consumption.

This chapter is organised, as follows. After a short introduction in Section 3.1, the basic model for a single server with PM is explained and elaborated in Section 3.2, followed by a description of the multiple server model in Section 3.3; both are analysed for various scenarios. Section 3.4 discusses the complexity and scalability of the approach, and Section 3.5 presents related work. The conclusions are described in Section 3.6.

This chapter is based on the publication [B10].

3.1 Introduction

Recall that one of the ways to reduce the overall energy consumption in a data centre is the use of PM. The main goal of PM is switching into a lower power state, while trying to keep the performance intact. More precisely, *advanced* PM concerns switching between the global and sleep power states (cf. Section 2.3.2). Additionally, the cascade effect, as described in [56], which is the profit in the infrastructure that is obtained from reducing of the energy consumption of the IT equipment, strengthens the effect of efficient control strategies for PM. Hence, even small reductions in percentages of energy consumption through efficient PM may have a large impact on the reduction of CO₂ emission and decrease overall cost. In this chapter, the simplest PM strategies considers only turning off idle servers, which can already save up to 20 to 60 % according to [35], [130], [139].

Efficient PM strategies can be explored by simulating and analysing the performance of various scenarios. Most techniques for exploring efficient PM use monitoring and testing. However, this reduces the number of applicable scenarios, since unpredictable behaviour has a low probability of occurrence and some scenarios cannot be easily tested. So, simulation and analysis of models provide a low-risk method that often result in remarkable insights into the performance of these systems. *Stochastic Petri nets* (SPNs) form a modelling technique which is convenient for performance analysis of systems. In this chapter, various SPNs are introduced that model PM of servers in a data centre, and that can be efficiently analysed using numerical techniques; for background information on SPNs, we refer to [76]. For our analyses, we use a well-known software tool MÖBIUS [36] that allows to model *Stochastic Activity Networks* (SANs) [156], an SPN extension.

This chapter aims to obtain better insight in power consumption and performance in data centres design phases. The contribution of this chapter lies in presenting high-level theoretical models, that allow us to calculate both power consumption and performance for a given data centre configuration and a given stream of jobs. However, these high-level models might include aspects that can be considered unrealistic. The chapter therefore also shows that extending the models is easy, in order to provide more realistic results.

In this chapter, two SPN models for advanced PM in a data centre are analysed: (a) a basic model with a *single server*; (b) an extended model with *multiple servers*. Even though the basic model could be called unrealistic, since it has only one server, it is still useful, because it provides fundamental insight in the power and performance trade-offs.

3.2 Single Server Model

3.2.1 Basic model

The basic idea of our models is illustrated in Figure 1.3, which is based on [B9]. A data centre serves a job stream from the outside world, buffers the incoming jobs and subsequently schedules and executes them, on the basis of the job requirements and the system-internal state information that is available. What that state information exactly is, largely depends on the data centre. It might involve only information on job queue lengths or server utilisation, but can also include information on temperature and humidity in parts of the data centre, or information in networking bottlenecks, to give just a few examples.

3.2.2 Visualisation of stochastic Petri net models

We follow the definition and semantics of SPNs as supported by the software tool MÖBIUS [36], in which places are depicted as circles, the input and output arcs are depicted as arrows, the immediate and timed transitions are depicted as respectively black and white rectangles (cf. Figure 3.1). All (non-zero) initial markings are depicted inside the circle of a place. The rewards are depicted below the places. The firing rates are depicted below the transitions.

3.2.3 Single server stochastic Petri net model

Starting from the basic model for a data centre in Figure 1.3, an SPN model is depicted in Figure 3.2. In the model, jobs arrive to the server with rate λ via the transition Arrivals to the place Buffer. Notice that arrivals can only take place as long there is buffer capacity available, i.e., as long as there are tokens in place BufferCap. In that sense, the arrival process forms a truncated Poisson process. The server can be in four states: the server is either turned off (a token in place

Figure 3.1: The elements of an SPN with immediate transitions and reward-based functions to determine output measures associated with places.

Figure 3.2: Basic single server Petri net.

Off), booting (a token in the place Booting), processing a job (a token in the place Processing) or idle, i.e., not processing a job, (a token in the place Idle). The corresponding rewards $P_0 = 0$, $P_b = 200$, $P_p = 200$ and $P_i = 140$ signify the power consumption (in W) when a token is present in such a place. Once jobs arrive in the buffer the server needs to be booted or the idle server can directly start processing the job. The server is booted with a delay, which in our model is fixed to an exponentially distributed amount of time with mean 100s ($\alpha = 100^{-1}$ jobs/s). After booting, the server processes the job with rate μ , where the size of the jobs determines the delay. For instance, a job size can be a web-request ($\mu = 1.00$ jobs/s, mean duration 1 s), a database request ($\mu = 0.10$ jobs/s, mean duration 10 s) or an upload/download of a file ($\mu = 0.01$ jobs/s, mean duration 100 s). The power consumption rates of servers and the job sizes are taken from [13], [48] and [66]. After a job is processed, the server becomes idle. Then, either a new job is processed by the server or it is shut down after a idle time-out with rate β , i.e., as soon as a server becomes idle, a timer starts, that shuts down the server after, an average $1/\beta$ time is expired.

In order to allow for a numerical solution (which is faster than simulation) the model should have a finite state space. For this reason, the place BufferCap is added, which is connected to the Arrivals transition with an output arc and the immediate transitions Start and Setup with input arcs. By setting the initial marking of BufferCap to K tokens, we make sure that there are never more than K jobs in the system. By setting K to a large value, we make sure that this finite state space has an insignificant impact on the performance measures of interest;

we check this by making sure that the probability that place Buffercap would be empty, i.e., disabling the arrival stream of jobs, is very small in all our models. The computed probability throughout the chapter is always almost zero (at least up to four digits).

To keep the models relatively simple, we have to make various assumptions. For our models to be able to obtain the underlying Markov chain (which is automatically derived and can be solved numerically), only models with exponentially distributed firing times (next to so-called immediate transitions) are allowed. In more detail, the current models assume a Poisson job arrival process, a server that has a fixed power consumption for the power states and an service time durations that follow an exponential distribution (which depends on the processing speed of the server and size of jobs offered to the server). PM shuts down the server after a time-out expires. The value of the timer is drawn according to an exponential distribution. In practice, PM will work with deterministic timers, hence, our model is approximate in that sense. Our assumptions can be relaxed in various ways. For some of these relaxations, the underlying model is still a Markov chain, which can still be solved efficiently numerically. For others, e.g., for fully deterministic time-outs, we will need to resort to discrete-event simulation.

3.2.4 Power-performance trade-off

The PM strategy in this chapter allows to turn off idle servers and turn these servers back on, which can decrease the power consumption, but has often a negative effect on the performance, i.e., there is a so-called *power-performance trade-off*. In this section, the power-performance trade-off is explored with several performance measures, such as utilisation and mean response time. In [76] measures to obtain from SPNs are discussed, which can be applied for finding the mean response time and the throughput. The other performance metrics are found via SPN properties and reward variables. Assume that $\mathcal{J} = \{\text{Off}, \text{Idle}, \text{Processing}, \text{Booting}\}$, $j \in \mathcal{J}$ and $m(j)$ denotes the number of tokens in place j for the following metrics:

The **mean number of jobs** $E[N]$ in the system is sum of the expected number of tokens in places that represent a job in the system:

$$E[N] = E[m(\text{Buffer}) + m(\text{Booting}) + m(\text{Processing})],$$

where $E[N]$ can easily be computed from the SPN.

The **throughput** X of jobs in the system is equal to

$$X = \lambda \cdot \Pr\{\text{BufferCap} \neq 0\},$$

such that a stable system, i.e., with sufficient buffer capacity, has a throughput of λ jobs per second, otherwise the throughput is capped, since jobs that do *not* fit in the buffer are discarded.

The **power-state utilisation** is the expected number of tokens in a place corresponding to that power-state, which is computed as follows:

$$\rho_j = E[m(j)]. \quad (3.1)$$

The **mean response time** $E[R]$ is the mean delay a job perceives from the moment it enters the buffer until the time it is finished with processing in a server. It is computed via Little's law [116] using the average number of jobs in the system ($E[N]$) and the throughput of jobs (X), as follows:

$$E[R] = \frac{E[N]}{X}. \quad (3.2)$$

The **mean power consumption** of the server $E[P]$ (in W) is computed with a reward variable that depends on the state of the server, as follows:

$$E[P] = \sum_{j \in \mathcal{J}} P_j \cdot \Pr\{m(j) = 1\}. \quad (3.3)$$

3.2.5 Results

In this section, the utilisation and trade-off between the mean power consumption and mean response time are explored.

Table 3.1 shows parameters values (partially taken from Section 3.2.3) for the single server model; it refers to the set B which is taken such that a good spread of parameter values for β is found, as follows: $B = \{10^{-2} \cdot 1.28^i \mid i \in \mathbb{Z} \wedge i \in \{0, \dots, 19\}\}$.

The **mean power consumption** $E[P]$ (y -axis) against a changing β rate (x -axis) is depicted in Figure 3.3 (from bottom to top: 1.0 (orange), 0.1 (blue), 0.025 (green) and 0.01 (red)). For small β , i.e., a long time-out value for an idle server, the mean power consumption is large, since the server is nearly always on and only needs to be booted very rarely. When β increases, i.e., a shorter shut down time for an idle server, the mean power consumption decreases and appears to converge to a fixed value. This is the case where a server is shut down nearly every time a job has been processed. Furthermore, the curves show that more power is consumed on average with a larger job size, i.e., when μ is smaller.

Figure 3.4 shows the **mean response time** $E[R]$ against a changing β rate. The mean response times are rather large for one server due to the impact of

booting servers, caused by PM. For this reason, when β is low, the mean response time is also low and when β increases the mean response time grows to a higher value.

Power-performance trade-off. Figure 3.5 illustrates the trade-off that exists between the mean response time and the mean power consumption. The mean response time is depicted on the x -axis and the mean power consumption is on the y -axis, for four different service rates μ . For each service rate μ , we clearly see that we can trade a lower mean power usage (to the left on the x -axis) for a higher mean response time (go up on the y -axis).

Note that the curve for $\mu = 0.01$ is much steeper than the other curves. This shows that the impact of changing β is relatively high for the mean response time and relatively low on the mean power consumption, compared to other

Parameter	α	β	λ	μ	K
Assigned Value	0.01	$\beta \in B$	0.007	0.01, 0.025, 0.1, 1.0	300

Table 3.1: Parameter assignments for **single server model**.

Figure 3.3: Mean power consumption against β , using the simple data centre model, for four different service rates.

Figure 3.4: Mean response time against β , using the simple data centre model, for four different service rates.

Figure 3.5: Parametric curve for trade-off between mean response time and mean power consumption, using the simple data centre model, for four different service rates.

service rates. The reason this curve is much steeper is that the number of jobs waiting for a server in the buffer is larger due to the relatively low processing rate ($\mu = 0.01$); this results in higher mean response times, especially when the server needs to boot more often. Hence, in this case the smallest increase in performance per Watt is obtained for shutting down a server.

3.3 Multi-Server Model

3.3.1 Stochastic Petri net model

The next step is to extend the single server Petri net model from Figure 3.2 with multiple servers, leading to the model illustrated in Figure 3.6. First, new servers are added to the model by increasing the number of tokens in place Off. So, the number of tokens $n \in \mathbb{N}$ in place Off is equal to the number of available servers.

The depicted model is very similar to an $M|M|m$ multi-server queue as in [76], where m represents the number of servers, with the difference that our model allows for the computation of power and performance for power management enabled servers. In such a multi-server queue, the arrival rate λ remains the same when adding servers, however, the service rate μ depends on the number of jobs i in the buffer, as follows:

$$\mu_i = \begin{cases} i\mu, & i = 0, 1, \dots, m \\ m\mu, & i = m + 1, m + 2, \dots \end{cases} \quad (3.4)$$

The same principle is applied to the rates of the transitions Delay, Service and Release in Figure 3.6, which are, respectively, $\alpha \cdot m(\text{Booting})$, $\mu \cdot m(\text{Processing})$ and $\beta \cdot m(\text{Idle})$, where $m(P)$ represent the marking in place P. Our model is structured in such a way that a new server can only be activated when there is a token in the place Off. So, there are always n tokens spread over the places Off, Booting, Processing and Idle. The arrival rate λ from transition Arrivals remains the same, since having multiple servers has *no* effect on this.

Note that the consequence of this approach is that PM is on a per-server level; there is *no* global timer for all the servers. So, each server starts a timer when it becomes idle, and turns itself off when the timer reaches its threshold, which is, as before, exponentially distributed with rate β .

In the current models, the servers are indistinguishable, since we assume homogeneous servers. As a consequence, all servers process jobs with rate μ , boot with rate α and release servers with rate β .

Figure 3.6: Multiple server extension of basic Petri net.

Furthermore, the power rewards now also depend on the marking. Obviously, if two servers are booting, i.e., two tokens are present in the place Booting, the power consumption for booting should be doubled. Note that in the model all servers have the same (distribution for) the booting time and for the processing jobs. So, for the places Booting, Processing and Idle the power rewards are adjusted to respectively $200 \cdot m(\text{Booting})$, $200 \cdot m(\text{Processing})$ and $140 \cdot m(\text{Idle})$. The power reward for a server that is off does *not* change, since the power consumption of turned off servers is exactly 0 W.

3.3.2 Power-performance trade-off

All of the power and performance measures from Section 3.2.4 can again easily be computed. Additionally, the mean power consumption *per server* is also computed, which allows us to reason about each server individually and to make comparisons of various scenarios more informative.

As before, a large buffer capacity ($K = 300$) is chosen, such that the computed mean values remain accurate with larger numbers of jobs in the system.

3.3.3 Results

Two scenarios are elaborated by adjusting at most two variables in the model.

Scenario I.

The first scenario addresses the impact of scaling the number of servers in a data centre with and without PM. For Scenario I, Table 3.2 presents the used parameters. Every parameter is fixed, except for the number of tokens in place Off. In this scenario, the case with PM is compared to the case without PM (for which we remove transition Release from the model, so that the servers are always on).

Since the number of servers increases and the arrival rate is fixed, the processing utilisation $\rho_{\text{Processing}}$ is expected to drop. The cumulative utilisation plot, as depicted in Figure 3.7 and Figure 3.8, show from bottom to top the computed processing utilisation $\rho_{\text{Processing}}$ (blue), booting utilisation ρ_{Booting} (green), idle utilisation ρ_{Idle} (orange) and off utilisation ρ_{Off} (red), confirms this expectation. The 2-servers case (top-left) has $\rho_{\text{Processing}} \approx 50\%$ with PM and without PM and the 10-server case (bottom-right) has $\rho_{\text{Processing}} \approx 10\%$ with PM and without PM. The greatest impact of shutting servers down is found with the lower processing utilisation with PM. For instance, in the 10-server case, the servers are expected to spend $\approx 35\%$ of the time off, while still $\approx 55\%$ of the time is wasted on booting a server and waiting as an idle server on jobs. In contrast to the case without PM, where servers are always on, *no* servers are turned off or need to boot and spend those moments idle.

Next, the mean response time is depicted in Figure 3.9. Note that lines are added for better visibility. The plot shows the impact of PM on the mean response time for multiple servers. The 2-server case shows a mean response time of approximately 1.85 s with PM and 1.33 s without PM (top-left). The 10-server case shows a mean response time of approximately 1.25 s with PM and 1 s without PM (bottom-right). An interesting minimal mean response time is found with the 4-server case with PM of ≈ 1.16 s. When servers are booting and no servers are available to be booted or idle, new incoming jobs have to wait in the buffer for a server. Since booting takes 100 s and processing only 1 s on average, the impact on the mean response time is the greatest with small

Parameter	α	β	λ	μ	n	K
Assigned Value	0.01	0.005	1.0	1.0	2-10	300

Table 3.2: Parameters assignments for **multi-server model Scenario I**.

Figure 3.7: Cumulative utilisation plot **with** PM when scaling the number of servers.

Figure 3.8: Cumulative utilisation plot **without** PM when scaling the number of servers.

Figure 3.9: Impact of PM on mean response time for various number of servers.

Figure 3.10: Impact of PM on mean power consumption for various number of servers.

number of servers, which explains the high mean response time for 2 servers. In the plot, an increase in the mean response time is recorded from 4 to 10 servers. The reason for this is that servers are only shut down when the server is idle and *no* jobs are waiting to be processed. Recall from Figure 3.7, that the 4-server case spends the same amount of time per job on booting as processing, whereas the 10-server case spends much more time per job on booting compared to processing, i.e., the 10-server case is much more often in power-state Off than the 4-server case. Therefore, the relatively long booting time compared to a short processing time slightly increases the mean response time.

Having seen the mean response time, next the mean power consumption is considered in Figure 3.10. The mean power consumption for 2 and 10 servers is respectively 340 W and 1460 W, without PM, in comparison to the case with PM, where 2 and 10 servers respectively have a mean power consumption of 342 W and 939 W. The slightly higher cost with PM for 2 servers is caused by booting servers. However, the energy reduction, which is caused by turning off idle servers, increases with the number of servers and has a larger impact on the mean power consumption than booting servers. The power consumption is reduced when the number of servers is larger than 4 compared to the case where servers are always on for this workload. If you use 1 to 4 servers, it is more efficient to *not* use PM at all for this load in this scenario.

Scenario II.

During the peak hours of a data centre, the number of arriving jobs increases. In this scenario, the impact of β and the number of servers are discussed by comparing two small data centres, one with 5 servers and one with 10 servers, where in both data centres the mean inter-arrival time ($1/\lambda$) between jobs and release rate (β) varies. Table 3.3 shows all parameters of Scenario II, which all are fixed, except for λ , β , and the number of servers, where β is taken from B as follows: $B = \{10^{-5} \cdot 1.28^i \mid i \in \mathbb{Z} \wedge i \in \{0, \dots, 27\}\}$.

Figure 3.11 shows the trade-off between the mean response time and the mean power consumption for 5 servers, three job arrival rates, $\lambda = 0.5$ (left curve), $\lambda = 1$ (middle curve) and $\lambda = 1.5$ (right curve), and varying β . The utilisation is expected to be really low for 5 servers. The data points close to $\beta = 0$ are in the lower part of the curves, and the data points for larger β are in the top of each of the curves. Again, there is a power-performance trade-off, which can be regulated by β .

For instance, consider the curve for $\lambda = 0.5$ for 5 servers (red). Values for β higher than 0.001 ($E[R] \approx 1.8$ and $E[P] \approx 124$) are all bad choices for β ,

Figure 3.11: Parametric curve for trade-off between mean response time and mean power consumption per server, using the multiple server data centre model with 5 servers, for three different job arrival rates.

since the mean response time and mean power consumption both increase. However, for β below 0.001, the power-performance trade-off exists. In more detail, when β decreases the mean power consumption increases and the mean response time decreases. The other curves show similar behaviour, only the curve for $\lambda = 1.5$ (blue) does not show the power-performance trade-off any more, which basically suggests that with the current PM strategy it is better to always keep the servers on.

Figure 3.12 shows the same trade-off for 10 servers with similar curves. Note that λ is the same as with 5 servers, so the 10 server case has even smaller load. For this reason the mean power consumed per server is lower than in

¹CTDP = Computation time per data point.

Parameter	α	β	λ	μ	n	K
Assigned Value	0.01	$\beta \in B$	0.5, 1, 1.5	1.0	5, 10	300

Table 3.3: Parameters assignments for **multi-server model Scenario II**.

Figure 3.12: Parametric curve for trade-off between mean response time and mean power consumption per server, using the multiple server data centre model with 10 servers, for three different job arrival rates.

Property	Single-server	Multi-server (I)	Multi-server (II)
number of states	604	910 – 3586	1856 – 3586
CTDP ¹ SSG (in s)	0.04	0.07 – 0.27	0.15 – 0.30
CTDP ¹ LUD (in s)	0.01	n.a.	n.a.
CTDP ¹ SOR (in s)	0.03	0.05 – 0.70	0.06 – 15.30
Total CTDP ¹ (in s)	0.04 – 0.07	0.12 – 0.97	0.21 – 15.60

Table 3.4: Computation time and size of state space.

the case with 5 servers. Moreover, for $\lambda = 1.5$ the power-performance trade-off still exists as a consequence of these smaller load.

3.4 Computational Cost and Scalability

For the evaluation of all the models, the provided SPN is translated by MÖBIUS into an underlying Markov chain; the number of states and the time for *state-*

space generation (SSG) for various cases is depicted in Table 3.4. As can be seen, even though the models have thousands of states, generating these does not cost much time at all. The generated Markov chain is subsequently solved numerically using *Successive over-relaxation* (SOR). For smaller models, roughly, below 1000 states, a direct *LU-decomposition* (LUD), can also be used. As becomes clear from the table, the solution times are all very small, so that easily many scenarios can be studied. The longest computation times were reported for model instances with very small β , apparently leading to so-called stiff models. All MÖBIUS settings were kept default or as recommended; the accuracy was set to 9 decimal places. All measurements have been performed on a machine equipped with a 2.70GHz INTEL® CORE™ i7-4800MQ CPU, 8 GB of RAM and WINDOWS 7 64-bit.

Having seen the computation times and sizes of the state space for Scenario I (Section 3.3.3) and Scenario II (Section 3.3.3), the size of the state space for Scenario I is computed to examine how scalable the models are, as follows:

Figure 3.13 shows the effect of scaling the number of servers on the number of states in MÖBIUS. The 10-server case shown (bottom-left) has 3586 states and the 100-server case (top-right) has 207151 states.

As a consequence of scaling the number of servers, the number of tokens in buffer capacity often needs to be adjusted when the arrival rate is increased. Figure 3.14 shows the effect of scaling the number of tokens for the buffer capacity for the 10-server case from Scenario II.

3.5 Related Work

The work of [69] focuses on power-performance trade-offs with PM, similar to this chapter. Also, PM is conducted in [102], which focusses on power-saving algorithms with hysteresis for adapting to the load of the system. In contrast to our work, both papers elaborate their models for virtualisation and/or multiple servers directly at the level of *Markov chains*, while we propose simpler SPN models, which are more generally applicable to data centres.

A similar remark applies to the papers [30] and [29], which both focus on modelling advanced server consolidation through virtualisation with the aid of numerical analysis of *Stochastic Reward Nets*. Their main goal is to propose and analyse virtual resource allocation strategies, which differs from our approach that focuses more on analysis of the power-performance trade-off for data centres with advanced PM.

Power-performance trade-offs for various policies are also explored in [64],

Figure 3.13: Number of states when scaling the number of servers from 10 to 100 servers computed with MÖBIUS.

Figure 3.14: Number of states when scaling the buffer capacity from K=300 to K=3000 computed with MÖBIUS.

in which an exact analysis of an $M|M|k|setup$ multi-server queue (k servers and a *setup* time) using the new *recursive renewal reward* (RRR), for solving Markov chains with repeating structures, is applied. They focus on demonstrating their proposed RRR technique with a data centre case study, while our focus is on the analysis of power-performance trade-offs with advanced PM.

The papers [15] and [118], which are related to the *All4Green* project, explore performance trade-offs at the server level, such that hardware components and energy-aware mechanisms for these components are taken into account. Other approaches, such as [27] and [26], focus on uncovering fundamental trade-offs (power, capacity, performance and dependability) with *Disk PM*, which is used for energy-aware file and storage systems. Also, here models for the power state of the system are analysed to discuss these trade-offs. In contrast to their work, our SPN models are defined on a much higher level of abstraction, such that the overall impact on power-performance can be discussed.

Our approach differs from all the above in proposing a flexible set of convenient and extendible SPN models, which are numerically solvable via MÖBIUS in order to analyse important power-performance trade-offs. Furthermore, the models are useful for data centres during the design phase to provide insight into trade-offs. In short, this chapter differs from other works in the application of an other modelling technique and/or aim for a different goal.

3.6 Conclusions

In this chapter, simple models for single server and multiple servers data centres with PM are analysed, to study the power-performance trade-off. For that purpose, SPN models have been defined from which we can easily and quickly derive important performance and power-usage measures, such as utilisation, mean response time and mean power consumption. To do so, we used the tool MÖBIUS. Interesting trade-offs between the mean power consumption and the mean response time are presented, which show that with PM reduction in the power consumption can be obtained at the cost of a higher mean response time.

For data centre analysts, an advantage of our approach is that the proposed models are reasonably high-level; this allows them to easily describe different configurations. Furthermore, the available analysis tools allow for the easy computation of relevant power and performance measures, thereby hiding mathematical details from the analysts.

Simulation Models

In this chapter, a simulation framework is proposed that allows for the analysis of power and performance trade-offs for data centres that save energy via advanced power management (PM). The models are cooperating discrete-event and agent-based models, which enable a variety of data centre configurations, including various infrastructural choices, workload models, (heterogeneous) servers and PM strategies. The capabilities of our modelling and simulation approach is shown with an example of a 200-server cluster. A cross-model validation that compares our results, for a restricted model with the numerical models, is also provided.

The chapter is organised, as follows. First, an introduction is provided in Section 4.1. The data centre and its context are then described in Section 4.2. Section 4.3 continues from this system description by introducing all models, metrics and visualisation. The case study with a 200-server example and cross-model validation are presented in Section 4.4, followed by Section 4.5 with the conclusions.

This chapter is based on the publication [B7].

4.1 Introduction

This chapter aims to obtain insight in power usage and system performance (measured in terms of response times) in early design phases of energy-efficient data centres. We take one of the energy-saving measures, viz., advanced PM (cf. Section 2.3.2), into account, which tries to lower power states of servers, while performance is kept intact. Moreover, the so-called cascade effect (to be discussed later; cf. [56]) on energy consumption in infrastructure, strengthens the effect of energy savings (by advanced PM in this case) made at server-level.

This chapter presents high-level models to estimate data centre power consumption and performance. We will present and simulate cooperating models for (i) IT equipment, (ii) the cascade effect, (iii) the system workload, and (iv) advanced PM. The value of our models is shown through the analysis and simulation of an example data centre. Our models combine *discrete-event* models and *agent-based* models. Simulating these models sheds light on power-performance trade-offs that are caused by the above-mentioned energy-saving advanced PM measure. For the construction of our models, the multi-method simulation tool ANYLOGIC [8] is used. ANYLOGIC supports a mixture of three common methodologies to build simulation models: (i) *system dynamics*, (ii) *process-centric/discrete-event* modelling, and (iii) *agent-based* modelling. Discrete-event modelling is a suitable approach for the analysis of systems that encompass a continuous process, that can be divided into discrete parts. Each part is characterised by triggering an event. As [105, p.6] states about *discrete-event simulation* (DES):

Discrete-event simulation concerns the modeling of a system as it evolves over time by a representation in which the state variables change instantaneously at separate points in time. These points in time are the ones at which an event occurs, where an event is defined as an instantaneous occurrence that may change the state of the system.

Agent-based modelling allows to model individual behaviour to obtain global behaviour via so-called communicating agents. It allows to specify heterogeneous populations. As [105, p. 694] states about agent-based simulation:

We define an agent-based simulation to be a DES where entities (agents) do, in fact, interact with other entities and their environment in a major way.

This chapter contributes by taking the first steps towards accurate insight in both power and performance by presenting simple queueing models of IT equipment that are easy to extend and allow heterogeneity. Also, a model for the cascading effect is taken into account, and workloads can be based on general probability distributions or on measurement data. Moreover, the insight in power and performance has strong visual support for transient and steady-state analysis. Next steps that follow from this research involve refining and validation of models for more realistic case studies based on measurements and knowledge obtained from cooperation with the project partners.

Over the last few years, various authors have proposed models for analysis of the power-performance trade-off in data centres. Recall from Section 2.5, that numerical solutions have been proposed to compute power and performance for data centres based on Markov models [63], [69], [102], fluid analysis [162] and stochastic Petri nets [30], [92], Chapter 3. All these numerical approaches allow for the rapid computation of trade-offs, but are often limited in their modelling capabilities, thus leaving them useful for only few metrics under limiting assumptions. Simulation using ANYLOGIC, as we propose here, might be slower, however, it can handle a wider variety of data centres than numerical analysis and scales well to larger systems (as we will see).

4.2 System Description

Recall from Section 2.2 that a data centre typically consists of various infrastructural components in order to meet high quality standards. Therefore, the data centre has an infrastructure similar to the overview as shown in Figure 2.2 with typical components like Automatic Transfer Switches (ATSs), Uninterruptible Power Supplies (UPSs), Power Distribution Units (PDUs), servers, chillers, coolers, network equipment and devices for monitoring and control.

Through the network, the data centre becomes accessible from the outside world. The **workload** of a data centre is the amount of work that is required to be done by the data centre. The workload of a data centre is an important indication for functionality and efficiency. An indication of the workload in a data centre is the number of *jobs* per time unit that arrive via the network, together with the length (distribution) of the jobs. Jobs sent through the network arrive in a buffer of a dispatcher, that schedules the jobs. We assume that storage and network equipment guarantee negligible job losses in this buffer.

Energy consumption can be reduced in data centres in several ways [56]. One way is **power management** (PM), that aims to switch servers into a lower

power state to reduce power consumption, while performance is kept intact. The challenge with advanced PM is to minimise the number of idle servers, while at the same time preventing unacceptable performance degradation. With advanced PM, saving energy is sometimes at the cost of performance, and results in a trade-off. We will illustrate such trade-offs later in the chapter.

4.3 Data Centre Models

This section presents the *Data Centre Simulation Framework* (DACSIM). Section 4.3.1 presents an overview of all implemented agent-based models. These agent-based models are built from underlying queueing models, state-chart models and functions for analysis, which are detailed in Section 4.3.2-4.3.6. Section 4.3.7 presents the power and performance metrics. Section 4.3.8 discusses the estimation method used. The simulation setup is then described in

Figure 4.1: Logo of DACSIM: A Data Centre Simulation Framework.

Figure 4.2: All implemented agents in one UML diagram.

Section 4.3.9. The live values and visuals of obtained from the simulation run are elaborated in Section 4.3.10.

4.3.1 Model Overview

All relevant entities are modelled as agents, which enables easy extension towards heterogeneous entities. An overview of all agents is given in the UML diagram in Figure 4.2.

The MainMenu agent (light yellow) links to the agents PowerPerformance, Infrastructure and Configuration with visual representation of the results (green). The other agents, i.e., DataCentre, Cascade, Dispatcher, EnergySupplier, Traffic, Power Management, Servers and Jobs are the data centre models, including a visual representation (light blue). In the upcoming subsections, the models inside these agents are discussed. The models inside the agent-based models are queueing models, state-chart models and functions for analysis.

4.3.2 Server Performance Model

Jobs arrive in a queue from a dispatcher. The dispatcher decides to which server the jobs should be dispatched depending on the state information.

Figure 4.3 shows a dispatcher (leftmost box) at which jobs arrive in a FIFO buffer according to a general arrival process and are served to one of the M servers after injection of the job in one of the server queues and waiting for service there. The dispatcher schedules jobs according to the shortest queue next

Figure 4.3: Dispatcher and servers queueing models.

algorithm (as explained in Section 2.4). However, other job dispatching policies are possible as well (e.g. round-robin) and is applied in the later chapters.

In order to compute response times, the Dispatcher agent flags a job with a time stamp before it enters the queue of the dispatcher. When a job is finished it subtracts the current time stamp with that of the time stamp before entering the queue to compute a response time sample.

Each Server agent comprises a $G|G|1|\infty|\infty$ queue (in Kendall notation; for more details about this notation we refer to [76]) with FIFO buffer. The jobs from the dispatcher are injected and arrive at the server queue. At most one job at a time is served with a generally distributed service time (with mean value $1/\mu$). If a server has been switched off, then no jobs are routed to it.

The main reason for this modelling approach, instead of directly using an $G|G|M|\infty|\infty$ queue, is that many job dispatching policies based on the state information of the server can be implemented in this framework, and it also allows for heterogeneous servers.

4.3.3 Server Power Model

Figure 4.4 shows a *deterministic finite automaton* (DFA; of which the formalism was introduced in [81]) that is used in the framework to switch between three global power states inside each server (as can be found in the ACPI open standard [79]; cf. Section 2.3.2). All state transitions in this model are invoked by either advanced PM or by jobs entering or leaving the system. Combining the time spent in each state with the per-state power consumptions (rewards) allows for the computation of (average) power consumption and energy consumption. There are three important effects that occur when switching power states: (i) job processing suspends/continues, (ii) transition from one power state to another takes time and consumes power, and (iii) power consumption decreases/increases. The system is **on** in all states of the model except for the states of (Off) and as (Asleep). Note that the time spent in the states sl (Sleeping), id (Idle), wk (Waking), bt (Booting) and su (Suspending) is considered overhead and, therefore, should be minimised.

To be more specific, a *power state* of a server indicates how the server is used and how much power is consumed for that use. The server state can be described with this DFA that switches between the low power consuming inactive Asleep state and the high power consuming active states Idle and Processing, that is controlled by external agents via messages; as shown in Figure 4.4. Initially, the server is idle, i.e., the initial state is Idle. When the server is active, it can switch between the power state Processing (R(pc)) and Idle (R(id)). When

Figure 4.4: The power model for switching between three global power states **asleep**, **on** and **off** with labelled states and in each state with abbreviation i an indication of the power consumed $R(i)$.

Parameter	$R(pc)$	$R(id)$	$R(as)$	$R(of)$	$R(sl)$	$R(wk)$	$R(bt)$	$R(su)$
Assigned Value	200 W	140 W	14 W	0 W	200 W	200 W	200 W	200 W

Table 4.1: Data centre configuration **power** parameters in this chapter.

a server receives a sleep message, it first needs time to suspend the system in power state Sleeping ($R(sl)$). After a generally distributed time with mean $1/\alpha_{sl}$, the server is in power state Asleep ($R(as)$). Power state Waking ($R(wk)$), which takes extra time before the server starts processing the first job, i.e., after a generally distributed time with mean $1/\alpha_{wk}$ the server is back on. The cycle to shut down and boot a server follows the following sequence of power states: Idle ($R(id)$) $\rightarrow$ Suspending ($R(su)$) $\rightarrow$ Off ($R(of)$) $\rightarrow$ Booting ($R(bt)$) $\rightarrow$ Processing ($R(pc)$). The servers leave the power state Booting after a generally distributed time with mean $1/\alpha_{bt}$ and the power state Shutting Down after a generally distributed time with mean $1/\alpha_{sd}$.

So, formally the DFA for switching between power states consists of the 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where $Q = \{as, wk, pc, bt, sl, id, su, of\}$, $\Sigma =$

$\{\text{waitForJob}, \text{injectJob}, \text{wake}, \text{woken}, \text{sleep}, \text{sleeping}, \text{boot}, \text{booted}, \text{suspend}, \text{suspended}\}$, $\delta : Q \times \Sigma \rightarrow Q$ (cf. all transitions in Figure 4.4), $q_0 = \{\text{Idle}\}$, $F = Q$. Additionally, the state rewards is defined as a function $R : Q \rightarrow \mathbb{R}$.

State rewards and time spent in particular states are provided throughout the chapters in table format. The power consumption values as used in this chapter are taken from [64] and shown in Table 4.1.

The used power state model is highly abstract and could be refined, e.g., based on recent results for CPU-intensive workloads [94].

The currently implemented *job dispatcher* depends on the power state of servers. Initially, an idle server is selected uniformly at random. If no idle server is present, an off server is selected uniformly at random. Another variant of a dispatching mechanism is to inject a job in the server with the shortest queue. In case there are multiple shortest queues, a random server is chosen; such (and other) variants can all be easily implemented in our framework.

4.3.4 Cascade Effect Model

The cascade effect, as mentioned already in Section 4.1, occurs in many data centre infrastructure components that consume power based on server power consumption, e.g., the amount of cooled air required in a data centre is related

Figure 4.5: EnergyLogic's cascade effect model.

to the amount of heat produced by all power consuming components.

The model for the cascade effect in data centres from [56], as depicted in Figure 4.5, is used in the Cascade agent. For each unit of power used by the servers, other data centre infrastructure components (cf. Section 2.2.1), e.g., power distribution and cooling, “waste” power in a linear relation. Hence, energy savings at the level of the server has great impact on the overall energy usage. The Cascade agent computes the power consumption metrics via simple linear functions.

4.3.5 Workload

Based on the server performance model from Section 4.3.2, jobs enter a dispatcher in a $G|G|1|\infty|\infty$ queue following a generally distributed inter-arrival time. In ANYLOGIC, the most common probability distributions are pre-implemented functions, e.g., exponential, normal, uniform and Erlang. The agent Job is added to the buffer after an inter-arrival time based on a function call that generates a random variable for the specified probability distribution. Additionally, in combination with the Traffic agent, custom discrete and continuous probability distributions can be defined using, e.g., frequency tables or observed samples. In this chapter, we only discuss generally distributed times with time-constant means and jobs with fixed mean lengths, yet our simulation does allow time-varying means in order to support realistic time-varying workload with heterogeneous jobs obtained from measurements in data centres.

4.3.6 Power Management Strategies

Without application of advanced PM, the servers in the data centre are either processing or idle. Advanced PM, however, aims to switch servers into lower power states to reduce power consumption when the workload is low, while performance is kept intact. The PowerManagement agent has functions to decide when servers need to be put to sleep or even switched off, and when servers need to be switched on.

In order to demonstrate the capability of implementing strategies in our framework, two of the functions are illustrated here. Customers of data centres often demand a certain performance with a *Service Level Agreement* (SLA), e.g., the response time in a data centre should never exceed 25 ms ($R_{\text{thres}} = 0.025$ s).

The *threshold strategy* tries to stay as close to this response time as possible by putting servers to sleep until it gets too close to the threshold and servers are again woken. In more detail, the response time gets too close to the threshold

when the latest observed sample exceeds 80 % of R_{thres} . Servers are put to sleep when the latest observed sample is lower than 60 % of R_{thres} . In the subsequent chapters, we will also investigate more advanced threshold strategies, e.g., including hysteresis.

The aim of the *shut-down strategy* is to achieve a workload of all active servers that is equal to a pre-defined percentage, e.g., a server workload of 20 % means a server spends on average 20 % of the time processing, when jobs are equally scheduled among all servers. As a consequence, servers are shut down to achieve that goal. The only exception to this rule is when there are not enough servers in the data centre.

4.3.7 Power-Performance Metrics

Quantitative metrics are used to provide insight into power and performance in data centres. This subsection distinguishes three important metrics: (i) power consumption, (ii) response time and (iii) power-state utilisation. These metrics are now further elaborated below.

Power Consumption

An infrastructure component c has power consumption $P_c(t)$ (in Watt) at time t (in seconds). Power consumption $P_{\text{server}_i}(t)$ of server i depends on the server's power state. The total power consumption of K servers $P_{\text{servers}}(t)$ at time t :

$$P_{\text{servers}}(t) = \sum_{i=1}^K P_{\text{server}_i}(t). \quad (4.1)$$

The power consumption of other system components (like infrastructure), $P_{\text{other}}(t) = \sum_{j \neq \text{server}_i} P_j(t)$, where $j \neq \text{server}_i$ from all other components is computed through the cascade model. The total power consumption then equals the sum of power consumption by all components, i.e., $P_{\text{total}}(t) = P_{\text{other}}(t) + P_{\text{servers}}(t)$. The mean power consumption up to time t is computed as:

$$E[P_{\text{total}}(t)] = \frac{1}{t} \int_{x=0}^t P_{\text{total}}(x) dx. \quad (4.2)$$

Note that this integral is not explicitly computed, but that an efficient discretisation takes place. This discretisation takes full advantage of the fact that events trigger changes in the power consumption, i.e., there is a piecewise

linear function for the power consumption over time. The mean power consumption up to time t , where k events occur at time $e_0, e_1, \dots, e_k$ within the interval $[0, t]$ with a fixed first event $e_0 = 0$ and a fixed last event $e_k = t$, is computed as:

$$E[P_{\text{total}}(t)] = \frac{1}{e_k - e_0} \sum_{i=0}^k \int_{x=e_i}^{e_{i+1}} P_{\text{total}}(x) dx \quad (4.3)$$

$$= \frac{1}{e_k - e_0} \sum_{i=0}^k (e_i - e_{i-1}) P_{\text{total}}(e_i). \quad (4.4)$$

Response Time

This is the delay R_i (in ms) from the moment a job i enters until the moment it leaves the data centre. So, each job will report its response time R_i . Given m observations, the mean response time is computed as:

$$E[R] = \frac{1}{m} \sum_{i=1}^m R_i. \quad (4.5)$$

Power State Utilisation

The power state utilisation $\rho_i(t)$ is the percentage of servers in a particular power state i at time t , with $\rho_i(t) \in [0, 1]$. The sum of all power state utilisations at time t is exactly 100%, i.e., $\sum_i \rho_i(t) = 1$.

The mean power state utilisation up to time t is computed as:

$$E[\rho_i(t)] = \frac{1}{t} \int_{x=0}^t \rho_i(x) dx. \quad (4.6)$$

In practice, the integral is not explicitly computed, but an efficient discretisation takes place, similar as done for the mean power consumption. The mean power state utilisation up to time t , where k events occur at time $e_0, e_1, \dots, e_k$ within the interval $[0, t]$ with a fixed first event $e_0 = 0$ and a fixed last event $e_k = t$, is computed as:

$$E[\rho_i(t)] = \frac{1}{e_k - e_0} \sum_{i=0}^k \int_{x=e_i}^{e_{i+1}} \rho_i(x) dx \quad (4.7)$$

$$= \frac{1}{e_k - e_0} \sum_{i=0}^k (e_i - e_{i-1}) \rho_i(e_i). \quad (4.8)$$

4.3.8 Estimation Method

The mean power consumption ($E[P]$) and mean response time ($E[R]$) are computed using the *batch means method* (BMM) [39], [62], [106]. The BMM subdivides the model time line into n equally large batches of which the averages $y_1, \dots, y_n$ are computed, and then computes the mean μ , standard deviation σ and 95% confidence interval CI of those computed averages. The BMM often requires the *model time* (t_{sim}) to be very long. Furthermore, observations are only recorded after a necessary transient start-up phase of the simulator, also called the *warm-up period* (wup). As details were missing for the estimation method used for discrete-event simulation within ANYLOGIC, the DACSIM implementation includes the BMM that extends the ANYLOGIC functionality.

To be more precise on the computation of all these values with the BMM, each average y_i is computed for the interval of its batch i , where $1 \leq i \leq n$, with the aid of the formulas from Section 4.3.7, i.e., the formulas are evaluated at the end of every interval and then subtracted from the previous evaluation to obtain the averages. While each average y_i is recorded during simulation, the mean μ , standard deviation σ and 95% double-sided confidence interval CI with corresponding t value ($n - 1$ degrees of freedom) from the Student distribution are calculated at the end of the simulation for each metric, as follows:

$$\mu = \frac{\sum_{i=1}^n y_i}{n}, \quad (4.9)$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (y_i - \mu)^2}{n - 1}}, \quad (4.10)$$

$$CI = \left[\mu - \frac{t\sigma}{\sqrt{n}}, \mu + \frac{t\sigma}{\sqrt{n}} \right]. \quad (4.11)$$

The computed averages $y_1, \dots, y_n$ are assumed to be independent and follow a normal distribution for estimation of the mean, variance and confidence interval. In order to ensure a good estimation, the results obtained with batch size n are compared to values obtained from $\frac{n}{2}$ and $\frac{n}{4}$ batch sizes to ensure insignificant differences in the final estimates. In order to ensure the right number of batches, this study could be extended to perform the Von Neumann test of independence [134] and the chi-square test of normality, as suggested by [34], to ensure the accuracy of the computed averages estimates.

4.3.9 Simulation Setup

DACSIM depends on the multi-method simulation tool ANYLOGIC and is required for proper execution of the simulation framework. For analysis of power and performance in DACSIM, the relevant settings made in ANYLOGIC and extensions made to ANYLOGIC are provided below.

In ANYLOGIC, discrete-event simulation is performed with a simulation engine that schedules events on a time line and adds new events that are generated by former events. By doing so, a chain of events is created. In order to provide the right accuracy with still good performance in DACSIM, the error margin for numerical methods used for mathematical equations is set to 10^{-4} by default and the time step used for fixed integration steps for these numerical methods is set to 0.1 s by default. For the purpose of reproducibility of simulation runs, most of the runs have a fixed random seed that has been set to an arbitrary value 314159265 by default for the random number generator. The model time unit is set to seconds, which will affect expression of rate, e.g., jobs per second. The number of batches n are set to $n = 50$ by default.

ANYLOGIC allows to set up a predefined customisable simulation experiment. DACSIM uses a standard *simulation experiment* for live results, e.g., plotting power and performance graphs (see Section 4.3.10). As a consequence, the CPU ratio (Presentation : Simulation), i.e., the ratio between a smooth presentation and a fast simulation, is set to vary between 1 : 2, for analysis with smooth visuals, and 1 : 16, for analysis with small breakpoints. Moreover, DACSIM uses a so-called *parameter variation experiment* for the rapid computation of multiple simulation runs in parallel for models with different parameters. The maximum memory usage is set to 4096 Mb by default for all simulation experiments.

All simulations have been performed on a machine equipped with a 2.70 GHz INTEL® CORE™ i7-4800MQ CPU, 8 GB of RAM and WINDOWS 7 64-bit with ANYLOGIC v8.1.0. The source code and executables of DACSIM are publicly available through the GITHUB repository [146].

4.3.10 Visualisation

The PowerPerformance and Infrastructure agents are implemented to show visuals and “live” values obtained from the simulation runs.

Figure 4.6 shows an intuitive dashboard with results and configuration parameters of the data centre model. The top line shows a menu bar with (1) links to the model, visuals and configuration. A cumulative utilisation plot

Figure 4.6: The dashboard for the IT equipment.

(2) shows “live” how many servers are in each power state. A stack chart below this plot shows the mean cumulative utilisation, i.e., how many servers are in each power state. Furthermore, two time plots (3) show “live” power consumption (left) and live response time (right) of the simulation. Two histogram plots (4) show the distribution of samples used to compute the means of power consumption (left) and response time (right). The values of the means are displayed in a small table including confidence intervals (5). Table (6) shows the exact number of servers in each power state, the total number of servers in the data centre and the total number of jobs in the queue(s). Configuration options (7) can be used to change the behaviour of the simulation on the fly: adjusting the server workload, the advanced PM strategies, reset the averages and disable averages are the main configuration options. Additional configuration options are available in the Configuration agent, like changing the arrival, service, and booting time distributions.

4.4 Results

First, an example of a data centre with a 200-server computational cluster is elaborated to illustrate the capabilities of the simulation models in Section 4.4.1. Next, steps are taken for to perform a cross-model validation by comparison of the results obtained from simulation to results obtained from models that are solved numerically in Section 4.4.2.

4.4.1 Case Study: Computational Cluster

We address a data centre that needs to be installed with 200 servers. A *Service Level Agreement* (SLA) permits a response time of at most 25 s. Jobs are served, and, require on average 1 s service time. Furthermore, we require that at most 33 % of all servers are processing, which is not unusual [24]. Booting and shutting down of servers require exactly 100 s and going to sleep and waking up need only 10 s. The *Power Usage Efficiency* (PUE) of the data centre is 1.5, i.e., 1 W saved at server level corresponds to 1.5 W saved in total; this is in line with the cascade effect model of Section 4.3.4. Furthermore, all the other IT equipment (that is, the non-servers) consume 1000 W, in total.

Table 4.2 shows an overview of distribution of the inter-arrival time between jobs (λ = arrival rate), service time distribution (μ = service rate), IT equipment specifications (mean booting time α_{bt} , mean shutting down time α_{sd} , mean sleeping time α_{sl} and mean waking time α_{wk} of servers), number of servers (n). Figure 4.7 shows the power consumption in each power state, combined with a legend for time-cumulative utilisation plot for the shut-down strategy.

First assume that the exact workload is known at all times, and the shut-down strategy (as described in Section 4.3.6) is applied. Figure 4.8 shows transient behaviour in a time-cumulative utilisation plot. The x -axis represents the model time t (in s) and the y -axis shows the percentage of servers in each of the power states. The workload without PM is around 33 %. With PM switched on, 50 % of all servers is shut down, such that 66 % of all active servers are processing jobs. Furthermore, the mean power consumption is ≈ 18 kW and the mean response time is ≈ 1 s.

In practice, the future workload is *not* exactly known. If workload prediction is inaccurate, late response of the PM strategy can dramatically increase the number of jobs in the system. Such situations have led to worse performance, either by dropped jobs or large queues.

The threshold strategy (as described in Section 4.3.6) is based on response times rather than on the workload to control the power state of servers. For

Parameter	λ	μ	α_{bt}	α_{sd}	α_{sl}	α_{wk}	n
Assigned Value	$\exp(33.0)$	$\exp(1.0)$	$\det(100)$	$\det(100)$	$\det(10)$	$\det(10)$	200 servers

Table 4.2: An overview of the data centre configuration **performance** parameters for the computational cluster.

Figure 4.7: Legend and power consumption in power-states.

Figure 4.8: Time-cumulative utilisation plot with shut-down strategy with time (in s) on the x -axis and cumulative utilisation (in %) on the y -axis.

Figure 4.9: Time-cumulative utilisation plot with threshold strategy with time (in s) on the x -axis and cumulative utilisation (in %) on the y -axis.

Figure 4.10: Time-response time plot with threshold strategy with time (in s) on the x -axis and response time (in s) on the y -axis.

Figure 4.11: Time-power consumption plot with threshold strategy with time (in s) on the x -axis and power consumption (in W) on the y -axis.

Figure 4.12: Response time samples distribution with threshold strategy with response time (in s) on the x -axis and frequency (in %) on the y -axis.

Figure 4.13: Power consumption samples distribution with threshold strategy with power consumption (in W) on the x -axis and frequency (in %) on the y -axis.

this strategy, the mean values are computed and time plots are generated (as can be seen from Figure 4.9–4.10). The mean response time $E[R] \approx 23$ s and mean power consumption $E[P_{\text{servers}}] \approx 20$ kW.

Figure 4.10 shows a time-response time plot with again on the x -axis the model time t and on the y -axis a green line interpolating between the response time samples. A horizontal red line is drawn to indicate the response time threshold $R_{\text{thres}} = 25$ s. Moreover, Figure 4.11 depicts a time-power consumption plot with model time t on the x -axis and a blue line interpolating between power consumption $P_{\text{servers}}(t)$ samples on the y -axis. Figure 4.9 shows a time-cumulative utilisation plot, where the x -axis represents the model time t (in s) and the y -axis shows the percentage of servers in each of the power states.

As seen in Figure 4.9–4.10, servers wake (for $t \in [1120, 1140]$), because the observed response times are approaching the threshold. Therefore, power consumption increases from ≈ 20 kW to ≈ 25 kW and the response time decreases from ≈ 24 s to ≈ 21 s. The next step is to put servers to sleep again (for $t \in [1140, 1220]$), because the perceived response time is fine. As a consequence, response times increase again from ≈ 21 s to ≈ 23 s, but power consumption decreases from ≈ 25 kW to ≈ 15 kW.

4.4.2 Cross-Model Validation

For a simpler but similar model, numerical solutions using stochastic Petri nets have been presented in Chapter 3, to also compute mean response times and mean power consumption, again with the focus to analyse the power-performance trade-offs caused by advanced PM (but without response time

and power consumption distributions).

In this chapter, we compare the power-performance metrics obtained from our simulation data centre models to similar metrics found in the numerical approach, as shown in Chapter 3. Therefore, the data centre model is configured to exactly the same rates, advanced PM strategy, number of servers and job dispatching as with the numerical solution. While this validation covers only a few scenarios, this comparison shows us two things: (i) we show the proximity of the power and performance values by expressing models with the exact same data centre scenario, and (ii) we show the difficulty of representing the exact same data scenario as a consequence of (most likely) hidden tool semantics. These models should be used with caution concerning its accuracy, while remaining valuable for discovering trends and dynamics.

Table 4.3 shows an overview of the data centre configuration performance parameters and capacity. The Poisson arrival rate $\lambda = 1.0$ jobs/s, $\alpha_{bt} = 0.01$ servers/s, and $\mu = 1.0$ jobs/s. A special PM strategy is implemented with an exponentially distributed release time with rate $\beta = 0.005$ servers/s that determines the number of servers shutting down per second when idle; note that deterministic time-outs are *not* allowed in stochastic Petri nets, which explains why the time-out has been chosen like this. The number of servers is scaled from 2 to 10. The time spent on shutting down a server α_{sd} is considered to be included in β or negligible in the numerical approach (see Figure 3.6).

Figure 4.14 and Figure 4.15 show cumulative power state utilisation plots for the servers with the PM strategy, for respectively the SPN-based numerical analysis and our simulation. The x -axis represents the number of servers n and the y -axis shows the percentage of servers in each of the power states (from top to bottom: red = off, orange = idle, green = booting and blue = processing). The plots confirm each other as the plots approach similar shape, but different values; the plots are not completely the same, which is partly the case due to the fact that we run a stochastic simulation, which, in essence, is a statistical experiment. Another reason for the observed difference lies in the implementation of the job dispatcher: the SPN-based models use only one general buffer,

Parameter	λ	μ	α_{bt}	α_{sd}	β	n
Assigned Value	exp(1.0)	exp(1.0)	exp(0.01)	n.a.	exp(0.005)	2-10 servers

Table 4.3: An overview of the data centre configuration **performance** parameters for the model validation scenario.

Figure 4.14: Cumulative utilisation plot when scaling the number of servers for *numerical* analysis (see Figure 4.7 for legend).

Figure 4.15: Cumulative utilisation plot when scaling the number of servers for *simulation* (see Figure 4.7 for legend)

Figure 4.16: Mean power consumption for various number of servers for simulation and numerical analysis.

Figure 4.17: Mean response time for various number of servers for simulation and numerical analysis.

whereas our simulation models use a separate buffer per server.

Figure 4.16 shows the mean power consumption for various servers in a data centre. The x -axis represents the number of servers n and the y -axis shows mean power consumption (in W). The curves for simulation and numerical analysis have similar shape, but with different values, which range, respectively, from 353 W and 342 W for 2 servers to 982 W and 939 W for 10 servers.

Figure 4.17 shows the mean response times for various servers in a data centre with the PM strategy. The shape of both curves are again very similar, but with different values, which are for simulation and numerical analysis respectively from 5.2 s and 1.84 s for 2 servers to 2.19 s and 1.25 s for 10 servers.

Another reason for the different values in the curves, is that the model used for numerical analysis (cf. Chapter 3) has no dispatcher, but an implicit way for dispatching jobs. First, jobs are scheduled to a random idle server with both numerical analysis and simulation. Otherwise, the jobs are scheduled to a random off server. If all servers are booting or processing, numerical analysis keeps jobs in the buffer, while the simulation injects jobs in a random server.

Note that the 95% confidence intervals computed are very small due to relatively long simulation runs. Consequently, the visual representation of the confidence intervals would be invisible in the graphs presented.

4.5 Conclusions

The contribution of this chapter is the presentation of a new ANYLOGIC-based tool called DACSIM with an intuitive dashboard, which is effective for obtaining quick insights in transient and steady-state behaviour of heterogeneous data centre with any possible workload and PM strategies. Furthermore, the ANYLOGIC environment enables to easily extend, refine and adapt data centre models to many other scenarios.

Insight is obtained in the power and performance in data centres with varying number of servers, PM strategies and workloads. Relevant metrics are derived from the qualitative data centre demands, including power consumption, response time and power state utilisation. These metrics are estimated by gathering samples from a mixture of discrete-event and agent-based models for IT equipment, PM and workload, implemented in ANYLOGIC. Furthermore, a cascade model enables the computation of total power consumption. Our approach is illustrated with a 200-server case study. Furthermore, we have performed a cross-model validation that showed similar results for power and performance for models solved numerically to the models simulated.

Specification of Advanced Power Management Strategies

In the previous chapter, the flexible simulation framework DACSIM has been proposed for the trade-off analysis of power and performance in data centres. The framework is now extended with a versatile module to study the effect of advanced power management (PM) strategies based on both power and performance measurement data collected during system operation. The chapter includes examples of several PM strategies presented in the literature (and extensions thereof). For a typical workload and a small server cluster, our more sophisticated PM strategies show significant energy consumption reductions, while performance is kept intact and Service-Level Agreements violations are minimised.

The chapter is organised, as follows. First, an introduction is provided in Section 5.1. An overview of approaches to analyse PM in a data centre environment is provided in Section 5.2. In order to describe PM strategies effectively, Section 5.3 provides a way to specify such strategies. DACSIM is extended with a module to support these PM strategies in Section 5.6. The application of the specification is shown in Section 5.4 by elaborating several PM strategies. Section 5.5 shows the simulation in action with a typical data centre configuration and these PM strategies. Conclusions are provided in Section 5.7.

This chapter is based on the publication [B4].

5.1 Introduction

Recall from Section 2.3.2 that (advanced) PM focusses on the suspension of idle or underutilised servers. By modelling such techniques and studying its results, useful insights can be found concerning the effects on energy-efficiency and performance [101]. Section 4.3.6 already introduced several example strategies that concern PM decisions based on several state (information) variables. This chapter provides a structured way to describe such strategies. A language is developed that allows for the analysis of strategies using many state variables. The observable state (information) variables taken into account in this language are: (i) queue sizes, (ii) response times, (iii) power state utilisation, (iv) power consumption, (v) temperatures, and (vi) time spent in power states, (vii) arrival rates, and (viii) service capacity of currently active servers.

Our new specification technique has been implemented as a module in the simulation framework DACSIM for analysis of power and performance (as elaborated in Chapter 4). The module extends PM functionality and introduces an interface to describe multiple strategies. The approach in this chapter is much more general, in the sense that it allows to use many more state variables than only queue size and time spent idling.

In the literature [11], [63] for PM, three strategic approaches exist: (i) *predictive*, (ii) *reactive* and (iii) *hybrid*. Predictive approaches like [35] and [99] use exponentially weighted averages, moving window average, linear regression and auto-regression methods to predict future values to determine if servers should switch to another power state. Reactive approaches like [117] and [82] base their switching behaviour on queuing theory estimates of response times and CPU utilisation. These approaches react to changes in for instance workload [66]. Hybrid approaches like [63] combine the above, where predictive approaches are useful for long-term workload trends and reactive approaches are used for short-term workload trends. A more elaborate discussion on these strategic approaches can be found in [11]. The method proposed in this chapter allows to specify and experiment with all of these PM strategies (and variations) in a single framework in a unified way.

5.2 Overall Approach

For this system, we make use of the simulation framework DACSIM elaborated in Chapter 4. The difference with the PM strategy approach from Section 4.3.6 is that the functions for several PM strategy examples are now replaced with a

Figure 5.1: System overview.

more flexible and separate PM module that allows us to structurally describe a wider variety of PM strategies. It is separate in the sense that the data centre simulation can be executed with and without PM features.

Figure 5.1 shows that PM is performed by controlling quantities of the data centre by observation and computation. The initial state of the system consists of a set of parameters that describe the data centre configuration and its job characteristics. For the purpose of advanced PM decisions, Section 5.2.1 discusses the observable quantities and Section 5.2.2 describes the controllable parts. Finally, Section 5.2.3 describes some valuable quantities derived from the observable quantities.

5.2.1 Observable Quantities

The information of the data centre offered to the power management module are the observable quantities of the system. With a growing number of sensors that collect data, much more information can be used in decision making, as follows:

Power state utilisation (PU) describes the percentage of time spend in a particular power state.

Power consumption (PC) describes the power consumed by servers and the data centre infrastructure (in Watt). *Expected power consumption* ($E[P]$) is the mean power consumption of all servers. *Expected energy consumption* ($E[E]$) is the expected energy consumed by all servers over the duration of the simulation.

Response time (RT) is a measure of performance, that indicates the total time a job takes from the time point it enters the system, to the completion of that

job by the system. *Expected response time* ($E[R]$) is the mean response time of a job.

Temperature (TM) indicates the temperature of the servers (in degrees Celsius).

Traffic concerns the job arrival rate (AR) that provides an indication of the work offered to the system.

System service concerns queue sizes (QS) and service capacity of the active servers (CA) that provides an indication of the service provided to the jobs in the system.

5.2.2 Controllable Parts

The observable quantities can be used, after some computation to control parts of the data centre, as follows:

Power state switching is a PM feature, that allows to switch between power states for particular servers.

Job dispatching is another controllable part, which has great effect on the quality of a PM strategy.

Recall from Section 2.3 that, switching between power states is controlled at the level of the operating system with the aid of the open standard Advanced Configuration and Power Interface (ACPI). This allows to specify policies at server-level. Nowadays, network monitoring and control allows to also specify these policies for a network of servers by invocation of Wake-on-LAN (WoL) techniques introduced by Intel and IBM [38], [85]. We assume that our system has these features are present in the data centres being studied. Figure 5.2 shows the state model used to switch between power states inside a server. A more detailed description of the power models used is provided in Section 4.3.3.

Recall from Section 2.4 that, the way that jobs are distributed to the available servers has significant impact on the performance. We assume that our system has these features available to distribute jobs via a dispatcher with dedicated network equipment. A more detailed description of the performance models used is provided in Section 4.3.2.

The impact on power and performance as a consequence of job scheduling and power state switching, suggests that a tight coupling between these two is expected to improve the overall quality of a PM strategy. Therefore, PM strategies require access to relevant observable quantities (as seen in Section

5.2.1) that provide a good indication of power and performance from collected sensor data.

5.2.3 Derived Quantities

The following metrics are derived from the observable quantities and provide valuable insight into the performance of the data centre:

Overhead Ratio (OR) is an indication of efficiency, i.e., the mean power-state utilisation of the idle, booting, waking and suspending power states divided by the total power-state utilisation. In efficient PM strategies, time spent in those power states is minimised, i.e., power states are only switched when necessary. An indication of the overhead power states is provided in Figure 5.2, which in fact is the only difference compared to the models presented in Figure 4.4.

Service-Level Agreements violations (SLAv) is the percentage of jobs that violate a certain response time threshold as stated in a so-called *Service-Level Agreement* (SLA). The SLA contains the service demands from data centre customers. Counting the number of violations and dividing this number

Figure 5.2: Model for power state switching (equivalent to the model presented in Figure 4.4) with an indication of the *overhead* power states (with a dark red border).

over the total number of samples, estimates the percentage of jobs that violate the SLA. This method is used in practice by one of our partners.

5.3 Strategy specification

This section proposes a formal specification technique for describing PM strategies that follows the system description from Section 5.2. First, a general description is provided in Section 5.3.1. This section discusses so-called *satisfiers* and *constraints*, which are further elaborated in respectively Section 5.3.2 and Section 5.3.3. An illustrative example is elaborated in Section 5.3.4.

5.3.1 Three-Step Approach

A PM strategy describes a high-level plan to achieve certain goals (e.g., reasonable performance, a stable power consumption and an evenly distributed temperature over all servers) while operating. This plan checks every r seconds if servers should be switching from one global power state to another via so-called *satisfiers*. After these satisfiers have evaluated several global conditions, it has become known if servers may switch to this particular global power state (Step 1). However, not all servers are suitable to be switched into another power state. Think of servers that are already overheated, only sleeping for just a few seconds or have a buffer full of jobs. Therefore, the set of suitable servers is narrowed down with so-called *constraints*, i.e., each server that complies to the list of conditions may be switched to that particular global power state (Step 2). To make this more formal, a PM strategy Θ is defined as a 3-tuple:

$$\Theta = (G, \Phi_S, \Phi_C(s)), \quad (5.1)$$

where the vector $G = (g_1, \dots, g_n)$ contains all possible global power states, the vector $\Phi_S = (\phi_S^{g_1}, \dots, \phi_S^{g_n})$ contains satisfiers for each global power state to switch to, and the vector $\Phi_C(s) = (\phi_C^{g_1}(s), \dots, \phi_C^{g_n}(s))$ contains constraints for each global power state to switch to for any PM enabled server s . These satisfiers in Φ_S and constraints in $\Phi_C(s)$ are formulas constructed from atomic satisfiers $\mathcal{S}$ and atomic constraints $\mathcal{C}$, which is further elaborated in Section 5.3.2 and Section 5.3.3.

For each PM strategy Θ^i with label i , in this thesis its vector with satisfiers Φ_S^i and vector with constraints $\Phi_C^i(s)$ are explicitly defined. Moreover, each strategy has three possible global power states in G^i (cf. Figure 4.4) namely **asleep** (*as*), **on** (*on*) and **off** (*of*).

The servers eligible after two steps of the approach, provide us in some cases with a non-deterministic choice as only a maximum of n_g servers are allowed to be switched to global power state g . Therefore, n_g servers are usually chosen by picking n_g servers uniformly at random from the remaining servers obtained after a selection procedure with satisfiers and constraints (Step 3).

Algorithm 1 provides the pseudo-code of the three-step approach. The algorithm iterates over all the global power states (line 1), that are covered by this thesis, to determine which servers are switched to that particular global power state. The strategy Θ is used to determine which servers are eligible for switching to the particular global power states g using the satisfiers and constraints

Algorithm 1: Pseudo-code of the three-step approach that is executed once every r seconds.

Data: S : set with PM enabled servers, Θ : PM strategy with the ϕ_S^g satisfiers and ϕ_C^g constraints for each global power state g , n_g : maximum # of servers to switch to g .

Result: For each global power state g , the servers in the set S'_g , given that $S'_g \subseteq S$ and $|S'_g| = n_g$, are switched to global power state g such that each server $Si \in S'_g$ complies to the satisfiers ϕ_S^g and constraints ϕ_C^g .

```

1 foreach  $g \in \{on, of, as\}$  do
2    $S'_g \leftarrow \emptyset$ ;
3   if  $S.COMPLIESTO(\phi_S^g)$  then                                     /* Step 1 */
4     /* Reorder serverlist uniformly at random */
4      $S.REORDERLISTUNIFORM()$ ;
5     while  $n_g \neq 0$  do
6       foreach  $Si \in S$  do
7         if  $Si.COMPLIESTO(\phi_C^g)$  then                               /* Step 2 */
8            $S'_g.ADD(Si)$ ;                                           /* Step 3 */
9            $n_g = n_g - 1$ ;
10        end
11      end
12    end
13  end
14 end
15 return  $(S'_{on}, S'_{of}, S'_{as})$ ;

```

as described above (lines 3 and 7). The first step of the approach determines if for global power state g , servers should be switched to it (lines 3–13). In the second step of the approach, determines which servers are eligible (lines 7–10). The third step of the approach, is done by reordering the servers uniformly at random and picking servers until the maximum number to switch to g is reached (lines 4–12). A set of servers S'_g is created (at line 2), filled with the chosen servers (at line 8) and returned as feedback to the controllable power state switching part (in line 15).

5.3.2 Satisfiers

The satisfiers observe various quantities, namely:

QS	is the total number of jobs in the queue of the system,
PU	is percentage of time the server spends in a particular power state,
TO	is the global time spent in a particular power state,
RT	is the response time,
PC	is the power consumption,
TM	is the temperature,
AR	is the job arrival rate,
CA	is the service capacity of the currently active servers.

This last mentioned satisfier is used as a very simple method for workload prediction, which could be extended to more elaborated workload prediction algorithms, as described in [179]. These satisfiers check if certain thresholds, as stated in the SLA, are exceeded. To sum up, an atomic satisfier $\mathcal{S}$ can be one of the observable quantities mentioned above, as follows:

$$\mathcal{S} := \text{QS} \mid \text{PU}(\delta, \gamma) \mid \text{TO}(\delta) \mid \text{RT}(\gamma) \mid \text{PC}(\gamma) \mid \text{TM}(\gamma) \mid \text{AR}(\gamma) \mid \text{CA}(\gamma), \quad (5.2)$$

where state $\delta \in \{\text{as, wk, pc, bt, sl, id, su, of}\}$ and computation method $\gamma \in \{\text{ins, mavg, eavg, bavg, per}\}$. In general, variable γ indicates how the satisfiers are computed, which are: instantaneous (ins), moving averages (mavg), exponentially moving averages (eavg), steady-state averages computed with the batch means method (bavg), and percentiles (per). The window size of the (exponentially) moving averages is set to 1 s by default. The variable δ indicates the state

the satisfier computes it for, and it may take the following values, that correspond to states in Figure 4.4: Waking (wk), Booting (bt), Sleeping (sl), Suspending (su), Asleep (as), Off (of), Idle (id) or Processing (pc).

The global power states **asleep** (*as*), **off** (*of*) and **on** (*on*) use the formula $\phi_S^{g_i}$ to determine if n_g servers should switch to power state g each r seconds, where r is the recurrence time of this check. Formula $\phi_S^{g_i}$ shows the expressiveness including a negate ($\neg$) operator, conjunction ($\wedge$) operator, disjunction ($\vee$) operator and parentheses for observing quantities with the *satisfiers* in $\mathcal{S}$, as follows:

$$\phi_S^{g_i} := \mathcal{S} \sqsubseteq \rho \mid \neg \phi_S^{g_i} \mid \phi_S^{g_i} \wedge \phi_S^{g_i} \mid \phi_S^{g_i} \vee \phi_S^{g_i} \mid (\phi_S^{g_i}) \mid \text{true} \mid \text{false}, \quad (5.3)$$

where $\sqsubseteq \in \{\leq, <, =\}$ is a comparison operator, g_i is a global power state and the domain of ρ depends on its satisfier $\mathcal{S}$.

5.3.3 Constraints

The server-specific constraints have much in common with how satisfiers are expressed. The main difference is that all constraints are at the level of the server. The server-specific atomic constraints are:

- QS(s) is the queue size of server s ,
- PS(s) is the current power state of server s ,
- TO(s, δ) is the time spent in a particular power state δ of server s ,
- TM(s) is the temperature of server s .

So, an atomic constraint $\mathcal{C}(s)$ can be one of the observable quantities mentioned above, as follows:

$$\mathcal{C}(s) := \text{QS}(s) \mid \text{PS}(s) \mid \text{TO}(s, \delta) \mid \text{TM}(s), \quad (5.4)$$

where s is a server and $\delta \in \{\text{as, wk, pc, bt, sl, id, su, of}\}$.

When n_g servers are to be switched to power state g each server must comply to certain *constraints*. Formula $\phi_C^{g_i}(s)$ shows the expressiveness including negate ($\neg$) operator, conjunction ($\wedge$) operator, disjunction ($\vee$) operator and parentheses for the constraints from $\mathcal{C}(s)$ for some server s , as follows:

$$\phi_C^{g_i}(s) := \mathcal{C}(s) \sqsubseteq \rho \mid \neg \phi_C^{g_i}(s) \mid \phi_C^{g_i}(s) \wedge \phi_C^{g_i}(s) \mid \phi_C^{g_i}(s) \vee \phi_C^{g_i}(s) \mid (\phi_C^{g_i}(s)) \mid \phi_S^{g_i} \mid \text{true} \mid \text{false}, \quad (5.5)$$

where $\sqsubseteq \in \{\leq, <, =\}$, g_i is a global power state and the domain of ρ depends on its atomic constraint $\mathcal{C}(s)$. To be more precise, for (i) $\text{QS}(s)$ holds that $\rho \in \mathbb{N}$, (ii) $\text{PS}(s)$ holds that $\rho \in \delta$ and only $\sqsubseteq \in \{=\}$ is supported, (iii) $\text{TO}(s, \delta)$ holds that $\rho \in \mathbb{R}$ and only $\sqsubseteq \in \{\geq\}$, (iv) $\text{TM}(s)$ holds that $\rho \in \mathbb{R}$.

5.3.4 Illustrative Example

Lets consider an example consisting of three servers ($\text{srv}_i \mid 1 \leq i \leq 3$) of which the first server is on and the last two are off. The PM strategy has to decide which server is put to sleep when evaluating the satisfiers formula $\phi_S^{\text{as}} := (\text{QS} \leq 10 \wedge \text{PU}(\text{as}, \text{mavg}) \leq 0.5)$ and the constraints formula $\phi_C^{\text{as}}(s) := (\text{TM}(s) \leq 30)$. The satisfiers formula requires before putting any server to sleep, that the global queue size is lower than 10 jobs and at the same time at most 50 % of all servers is sleeping, determined with a moving average. The constraints formula narrows down the amount of servers, that can be put to sleep, to those servers, that have a temperature of at most 30 °C.

Suppose that, as depicted in Figure 5.3, at the moment of evaluation, temperatures for servers srv_i are respectively 24 °C, 34 °C and 20 °C, the total number of jobs waiting in the system is 8 and the percentage of time spent in the asleep state determined with a moving average is 20 %.

Figure 5.4 shows all possible next power states the server could be in when switching one or more servers to the asleep state. In this figure, states of the form (s_1, s_2, s_3) , where $s \in \{I, O, S\}$, signify the states in which the server is either on (I), off (O) or asleep (S). First, the satisfiers formula ϕ_S^{as} reduces the state space S to $S' = \{(I, S, O), (S, O, O), (I, O, S)\}$, because at most one server will be put to sleep, there are less than 10 jobs in the system and the percentage of time spent in the asleep state determined with a moving average is below 50 %. In the next step the formula $\phi_C^{\text{sl}}(s)$ reduces the potential future state space S' even further to $S'' = \{(S, O, O), (I, O, S)\}$, because $s_2 = (I, S, O)$ violates the constraints formula $\phi_C(s)$ with a temperature of 34 °C. After evaluation of the formulas, one of the remaining servers will be chosen uniformly at random.

5.4 Application

Using the specification presented above, we now describe five PM strategies that we intend to evaluate. First, a traditional energy-inefficient approach is elaborated as a base case PM strategy in Section 5.4.1. A connection to existing literature PM strategies is made in Section 5.4.2. Section 5.4.3 uses these as a starting point to propose more advanced strategies.

Figure 5.3: A single iteration of the three-step approach with a illustrative snapshot of the system state that are compared to the global-level satisfiers and local-level constraints (green=complies with formulas, red=violation formulas).

Figure 5.4: Switching three servers from a particular state to the sleep state using satisfiers and constraints.

5.4.1 Base Case Strategy: AlwaysOn

The strategy *AlwaysOn* (Θ^{all}) is defined to set a base case for discussion of the impact of using a power management strategy. The satisfiers and constraints formulas for Θ^{all} , with $G^{all} = (on)$, are as follows:

$$\Phi_S^{all} = (\phi_S^{on} := (true)), \Phi_C^{all}(s) = (\phi_C^{on}(s) := (true)). \quad (5.6)$$

Essentially, (5.6) decides that every server should be turned on whenever possible, i.e., the global-level conditions are always satisfied and all servers are allowed by the local-level conditions.

So, for this base case, all servers are considered to be on at all times. This is expected to lead to high performance, but also to high energy consumption for the job and data centre characteristics in Section 6.4.1.

5.4.2 Literature Inspired Strategies: Optimal and Demotion

In the literature various approaches can be found with regard to provisioning of servers in a dynamic way. In [99] and [35], request rates are predicted via last arrival, (exponentially) moving averages, linear regression and auto regression. Inspired by these approaches satisfiers formulas typically contain the arrival rate $AR(\gamma)$ satisfier. Even variants such as *SoftReactive* [66], that add time-outs before an idle server is allowed to be turned off, are covered with our time-out constraint $TO(s, \delta)$. In combination with the current power state $PS(s)$ constraint, that only allows servers to switch to a power state if it is being switched from a particular power state.

In [82], two PM policies are described for comparison: *Optimal*, which uses an algorithm to determine the deepest power states servers can be in without disturbing the service; and *Demotion*, which gradually demotes servers into deeper power states, i.e., switching into a power state that often consumes less power, but takes more time to be activated.

The *Optimal* policy computes the number of servers S_i^* that should be in each particular power state i by estimation of the response times. Additionally, wake up times are taken into account. The technique elaborated in their paper is not literally implemented, but could be added to γ . Inspired by this policy, we observe that estimation of response times ($RT(mavg)$) is sometimes used for determining a PM strategy. An example of a typical satisfiers formula, that belongs to our Θ^{opt} and Θ^{dem} strategies with $G^{opt} = G^{dem} = (as, on, of)$, are defined as follows:

$$\Phi_S^{opt} = \Phi_S^{dem} = \left(\begin{array}{l} \phi_S^{as} := (RT(mavg) \leq R_{SLA}) \\ \phi_S^{on} := (RT(mavg) > R_{SLA}) \\ \phi_S^{of} := (true) \end{array} \right), \quad \begin{array}{l} (5.7a) \\ (5.7b) \\ (5.7c) \end{array}$$

$$\Phi_C^{opt}(s) = \left(\begin{array}{l} \phi_C^{as}(s) := (PS(s) = id) \\ \phi_C^{on}(s) := (PS(s) = as) \\ \phi_C^{of}(s) := (false) \end{array} \right). \quad \begin{array}{l} (5.8a) \\ (5.8b) \\ (5.8c) \end{array}$$

According to (5.7a), servers are only put to sleep if the observed moving average response times of all servers has become sufficiently small, i.e., the moving average response times are compared to an average response time threshold R_{SLA} , that has been specified in the SLA. Similarly in (5.7b), servers are again woken if the observed average response times of all servers exceed the same average response time threshold R_{SLA} . For the second step of the approach, basic constraints select the servers that are idling, as shown in (5.8a), if servers can be put to sleep, and select the servers that are asleep, as shown in (5.8b), if servers can be woken. Although the PM strategy expects support of turning a server off, according to (5.7c) and (5.8c), this feature is unused, i.e., servers are never turned off.

The *Demotion* policy introduces time-outs to the idle machines (t_{id}) and sleeping machines (t_{as}), that should expire first before going into a deeper power state. Furthermore, it requires to be able to switch from the off power state to a sleep power state and vice versa. This would require the two additional states and four extra transitions in the state diagram from Figure 4.4 connecting the Asleep and Off state. This policy depends strongly on how job scheduling is implemented. For instance, when using shortest queue next for dispatching jobs, servers spend only little time in the Idle power state, assuming there are no specific server demands or priorities involved. The following constraints formulas describe such a type of policy:

$$\Phi_C^{dem}(s) = \left(\begin{array}{l} \phi_C^{as}(s) := (TO(s, id) \leq t_{id}) \\ \phi_C^{on}(s) := (PS(s) = as \vee PS(s) = of) \\ \phi_C^{of}(s) := (TO(s, as) \geq t_{as}) \end{array} \right). \quad \begin{array}{l} (5.9a) \\ (5.9b) \\ (5.9c) \end{array}$$

In the case of the Demotion policy, the same satisfiers as with the Optimal policy apply as described in (5.7). According to (5.9a), the servers, that are

selected in the second step of the approach to be put to sleep, are the servers, that have spent at least t_{id} s time idling. Servers are only switched from asleep to off if t_{as} s have expired, as shown in (5.9c). Furthermore, (5.9b) selects all asleep and off servers can be activated.

As can be seen from these examples, PM strategy variations are endless. Our approach helps us to categorize strategies and forces us to write down our assumptions in a strict and unambiguous way.

5.4.3 Fine Tuned Strategies: Strong and Advanced

We now extend the strategies Optimal and Demotion (as described in Section 5.4.2) as follows.

The *Strong* extension is inspired by running instances of the Optimal and Demotion variants. This showed that the strategy did not perform well with a small number of servers. Therefore, the Strong strategy never shuts down the last 20% of the total number of servers. In order to prevent overly active power state switching by the strategy, servers can only turn a server back on if it has been sleeping for at least 100 s. The satisfiers and constraints formulas for Θ^{str} strategy with $G^{str} = (as, on)$ are as follows:

$$\Phi_S^{str} = \left(\begin{array}{l} \phi_S^{as} := ((RT(eavg) \leq \frac{3}{4} \cdot R_{SLA}) \\ \wedge (PU(id, ins) \geq 0.2)) \\ \phi_S^{on} := (RT(eavg) > \frac{3}{4} \cdot R_{SLA}) \end{array} \right), \quad (5.10a)$$

$$(5.10b)$$

$$(5.10c)$$

$$\Phi_C^{str}(s) = \left(\begin{array}{l} \phi_C^{as}(s) := (PS(s) = id) \\ \phi_C^{on}(s) := (TO(s, as) \geq 100.0) \end{array} \right). \quad (5.11a)$$

$$(5.11b)$$

In (5.10a), servers are allowed to be put to sleep if an exponentially moving average response time is below $\frac{3}{4}$ of the response time threshold R_{SLA} and the currently observed power state utilisation of idle servers is greater than 20%. The response time threshold introduces a safety bound to prevent oscillating behaviour of PM strategies to cause SLA violations. Also, the *exponentially* moving average response time contributes by allowing a small safety bound, as it creates an even smoother course of the observed response time curve. If these exponentially moving average response time exceeds $\frac{3}{4}$ of the response time threshold R_{SLA} , servers are again woken, as described in (5.10c).

For the second step of the approach, the servers to be put to sleep must be idle according to (5.11a). The constraints from (5.11b) select only the servers that are asleep for period of 100.0 s or more.

The *Advanced* extension adds some additional efficiency. If the processing power state utilisation of a business data centre from Figure 5.5 is really typical, then 50% is expected to do nothing. So with a safety bound of at least 20% idle servers, at most 30% of all servers can be turned off. The satisfiers and constraints formulas for Θ^{adv} strategy with $G^{adv} = (as, on, of)$ are as follows:

$$\Phi_S^{adv} = \left(\begin{array}{l} \phi_S^{as} := (RT(eavg) \leq \frac{3}{4} \cdot R_{SLA}) \\ \quad \wedge PU(id, ins) > 0.2 \\ \phi_S^{on} := (RT(eavg) > \frac{3}{4} \cdot R_{SLA}) \\ \phi_S^{of} := (PU((id, ins) > 0.2) \\ \quad \wedge (PU(of, ins) \leq 0.3) \\ \quad \wedge (PU(bt, ins) \leq 0.05)) \end{array} \right), \quad \begin{array}{l} (5.12a) \\ (5.12b) \\ (5.12c) \\ (5.12d) \\ (5.12e) \\ (5.12f) \end{array}$$

$$\Phi_C^{adv}(s) = \left(\begin{array}{l} \phi_C^{as}(s) := (PS(s) = id), \\ \phi_C^{on}(s) := ((PS(s) = as \vee PS(s) = of) \\ \quad \wedge (\neg(PU(as, ins) > 0.0) \\ \quad \wedge (PS(s) = of)) \\ \phi_C^{of}(s) := (TO(s, as) \geq 100.0) \end{array} \right). \quad \begin{array}{l} (5.13a) \\ (5.13b) \\ (5.13c) \\ (5.13d) \\ (5.13e) \end{array}$$

The satisfiers (5.12a)–(5.12c) for switching servers on and asleep are equivalent to the satisfiers of the Strong strategy. Servers can only switch to the off states if the currently observed idle power state utilisation, i.e., the fraction of all servers that are idle, exceeds 20% in (5.12e). Also, for the servers to be switched off, the fraction of servers that are off must be below 30% in (5.12e) and the fraction of servers that is booting must be below 5% (5.12f) to prevent overly active power state switching.

The constraint (5.13a) is equivalent to (5.11a) of the Strong strategy. Similar to the constraint (5.11b) of the Strong strategy, the servers selected to turned off in (5.13e) are the servers that are asleep for at least a period of 100 s. For servers to turn on, the server should either be asleep or off is stated in (5.13b). In order to remain efficient, servers can only be selected that are turned off (5.13d) if

no server can be woken up from the asleep power state (5.13e), i.e., it is more efficient to wake up a server than to boot a server.

5.5 Results

To illustrate the full assessment of the qualities of a PM strategy, an implementation of various PM strategies and computation of their qualities together with a data centre configuration is elaborated in Section 5.5.1. Section 5.5.2 provides an example quality assessment of several strategies.

5.5.1 Data Centre Configuration

Table 5.1–5.2 show the parameters for an example data centre. These parameters are supposed to represent a common day for a given data centre. Jobs arrive in the load balancer according to a Poisson process with rate λ (job/s). The service rate μ (job/s) of each server is exponential, i.e., the server finishes jobs in a varying amount of time, because of varying job sizes. The model is extended to support four power states, as in Figure 4.4. The time required for a server to shut down α_{su} , to boot α_{bt} , to wake α_{wk} and to go to sleep α_{sl} are deterministic. The awareness of power states in the models allows to compute power consumption (P) by rewarding each power state with a power consumption (rate). Note that power state Processing is the only power state in which jobs are served.

Job dispatching is an essential part of the data centre configuration, since it influences the performance a lot. Jobs arrive at the data centre and are distributed via a dispatcher. The dispatcher uses the default scheduling of jobs

Parameter	R(pc)	R(id)	R(as)	R(of)	R(sl)	R(wk)	R(bt)	R(su)
Assigned Value	200 W	140 W	14 W	0 W	200 W	200 W	200 W	200 W

Table 5.1: Data centre configuration **power** parameters.

Parameter	λ	μ	α_{sl}	α_{wk}	α_{bt}	α_{su}
Assigned Value	exp(20.0)	exp(1.0)	det(10.0)	det(10.0)	det(100.0)	det(100.0)

Table 5.2: Data centre configuration **performance** parameters.

Figure 5.5: Daily processing utilisation of a typical business data centre [56].

via shortest queue next (cf. Section 2.4). However, jobs can only be dispatched to servers in the idle and processing power states. In order to achieve this, we used the module extension of DACSIM as proposed elaborated later on in Chapter 9.

The mean power consumption ($E[P]$) and mean response times ($E[R]$) are computed using the batch means method. The batch means method requires the model time (t_{sim}) to be very long, here $t_{\text{sim}} = 86\,400$ virtual seconds (exactly 1 day/night cycle), and the system should finish its start-up phase after some warm-up (wup) period. The total number of servers available (n) in the simulation is fixed to $n = 30$. The SLA response time threshold (R_{SLA}) is set to $R_{\text{SLA}} = 20$ s. Violations of this threshold could result in a penalty or are considered to be unacceptable.

Figure 5.5 indicates daily server utilisation of such a business data centre based on data from [56]. Since the processing speed of these servers remains the same, this server processing utilisation could be rewritten as a time-dependent Poisson process with a job arrival rate λ for all servers of $\frac{(\mu \cdot n)}{100} \cdot \text{PU}(\text{pc}, \text{ins})$, where $\text{PU}(\text{pc}, \text{ins})$ is the currently observed processing utilisation (in %) and $(\mu \cdot n)$ is the maximum allowed number of jobs arriving per second. This assumes that overhead caused by scheduling jobs is negligible.

The advantage of this approach is that data centres are simulated by only analysing its day-night patterns. This can be done by deriving the inter-arrival times of jobs for each hour together with the other job characteristics, such as priorities and size. Validation of such parameters is considered as future work.

5.5.2 Example Strategies

A global overview of the computed values (as discussed in Section 5.2.1–5.2.3) of all PM strategies (as seen in Section 5.4.2–5.4.3) is provided in Table 5.3. Each row in the table represents one PM strategy and its computed values. Beside the measurements itself, each value is ranked (from the highest (1) with a dark green background to the lowest (5) with a light green background) by comparing with the other strategies.

One of the relevant observations from Table 5.3 is the 54% energy reduction for the Advanced strategy (53.02 kWh for 1 day) compared to using AlwaysOn strategy (115.10 kWh for 1 day). However, such an energy reduction caused by PM has some effect on other quantities for the data centre. Optimal and Demotion both show already a large improvement in energy efficiency with reasonable performance. With some additional fine tuning, Strong boosts the performance and reduces the number of SLA violations. Advanced has even less energy consumption with still fine performance.

Strategy	E[E] (in kWh)		E[P] (in W)		E[R] (in s)		OR (in %)		SLAv (in %)	
Base Case										
AlwaysOn	(5)	115.10	(5)	4796	(1)	1.269	(5)	66.86	(1)	0
Literature-based (inspired by [82])										
Optimal	(4)	59.09	(4)	2462	(5)	8.761	(3)	4.69	(5)	2.57
Demotion	(2)	55.06	(2)	2294	(4)	8.665	(4)	4.78	(4)	1.879
Fine-tuned										
Strong	(3)	56.14	(3)	2339	(3)	4.257	(1)	3.59	(2)	0.0021
Advanced	(1)	53.02	(1)	2209	(2)	4.101	(2)	3.90	(3)	0.0130

Table 5.3: Power management strategies analysis.

5.6 Power Management Module Implementation

As seen in Figure 5.1, power management can be seen as an independent part from the rest of the data centre. This structure allows us to extend existing work (cf. Section 4.3.1) by adding a module. Figure 5.6 shows how the simulation framework is extended with this PM module. Thermal-awareness, load balancing policies and PM modules are loaded in the framework by creating an instance of the DataCentre class. Therefore, the DataCentre class has an instance of the PM module ModPowerManagement class.

The ModPowerManagement class describes all the power management strategy satisfiers $\phi_S^{g_i}$ and constraints $\phi_C^{g_i}(s)$. Furthermore, the ModPowerManagement loads one power management strategy that complies to the PMStrategy interface. This interface requires to specify $\phi_S^{g_i}$ and $\phi_C^{g_i}(s)$ that are used to switch to the global power states with the three functions checkGotoOn, checkGotoOff and checkGotoSleep. Each implementation of PMStrategy has these functions to ensure specification of $\phi_S^{g_i}$ and $\phi_C^{g_i}(s)$.

Figure 5.7 shows an UML class diagram that gives an overview of the implementation of the two power management module classes ModPowerManagement and PMStrategy. The function remoteManagement is called recurrently after time r from the function getRecurrenceTime. This function activates the check... functions to check the satisfiers and constraints via the sat... and cons... functions. Recall that the satisfiers and constraints leaves the system with a set of allowable servers to switch to this power state. This set of allowable servers is often larger than the number of servers that will be switched on, so the strategy how to pick these servers and the communication back to the simulation framework is elaborated in the remoteGotoSleep, remoteGotoOff and remoteGotoOn functions.

The underlying ANYLOGIC simulation tool of the data centre simulation framework DACSIM has an option for a (i) *live simulation*, that has a graphical user interface and live graphical output, and a (ii) *parameter variation experiment*, that has quick and parallel execution of simulation runs without the live features. Both options have been implemented, because live simulations are useful for quick insight into strategy behaviour and parameter variation experiments are rather useful for assessment and generation of relevant data. As a consequence, an additional class has been implemented to export all relevant power management values to a comma-separated values (CSV) or EXCEL file.

Additionally, a special class ParamVariationPMStrategy has been implemented for parameter variation experiments to explore larger sets of design. This class complies to the PMStrategy interface, and needs to be instantiated with

Figure 5.6: High-level design UML Class Diagram structure for the PM module.

Figure 5.7: Low-level design UML Class Diagram for two main PM classes.

a large number of parameters. With these parameters many formulas, as described in Section 5.3, are covered.

Small changes are made within the simulation framework core by adding some measures to the Dispatcher class to obtain SLA violations and the OR. Inside the Server class, the ability to switch from off to sleep and vice versa, has been implemented as well. This extension is used for strategies that put servers into a deeper power state.

5.7 Conclusions

In this chapter, we propose PM strategies to be specified to meet data centre quality demands for some global and server-level conditions. The specification allows to structurally specify state information variables of a data centre and its usage in advanced PM. Furthermore, a bridge between existing literature and this new specification is made to show its general applicability. Moreover, it has shown to be useful for categorising scientific contributions in the field of advanced PM.

A module to the existing DACSIM simulation framework allows to easily compare these PM strategies as defined in the specification.

A small 30 servers cluster with typical small business data centre workload is analysed for power and performance. An energy reduction of 54% is obtained by PM strategies inspired by the literature (Optimal and Demotion) and fine tuning thereof (Strong and Advanced). Energy-efficiency is significantly improved, while performance is kept intact and Service-Level Agreement violations are minimised.

Evaluation of Advanced Power Management Strategies

In this chapter, we study the quality of advanced power management (PM) strategies based on both power and performance measurement data collected during system operation. These strategies take a wide variety of state variables into account. In order to ensure the quality of new strategies, they are studied for stability, efficiency, adaptability and robustness; these qualities will be formally defined. This chapter presents an evaluation approach for these qualities, and demonstrates the approach for several PM strategies, as introduced in Chapter 5. We show that the choice of PM strategy depends both on which qualities are given the highest priority and on the used state information.

The chapter is organised, as follows. First, an introduction is provided in Section 6.1. An overview of the approach is provided in Section 6.2. Section 6.3 describe strategy qualities that are helpful to find the best and most suitable strategy for a given data centre configuration. Section 6.4 shows the evaluation in action with a typical job and data centre configuration and several interesting PM strategies. Conclusions are provided in Section 6.5.

This chapter is based on the publication [B2].

6.1 Introduction

Just like in the previous chapter, we discuss advanced PM that focusses on the suspension of idle or underutilised servers. Our new evaluation method is embedded in our work from Chapter 5 that introduced specification and module extension in the simulation framework DACSIM (from Chapter 4). The module extends PM functionality by proposing an interface to easily specify multiple strategies. Moreover, such a specification allows to use many state information variables with regard to traffic, system service, power, performance and thermodynamics, to formulate all kinds of more advanced PM strategies/policies that could lead to significant improvements in energy-efficiency, while other Service-Level Agreement (SLAs) demands are met.

This chapter uses this specification method to study multiple PM strategies and proposes a set of metrics to evaluate the quality of advanced PM strategies simulated in our framework. We study PM strategies for efficiency and stability including minor variations (robustness) and the impact of adapted workloads (adaptability); these four quality measures will be formally defined.

In this chapter, an evaluation example is elaborated that assesses the quality of five PM strategies from Section 5.4 that are based on the literature and fine tuning thereof. The evaluation in this chapter is more extensive than the example presented in Section 5.5.2 by providing more and roughly categorised metrics in one of the four quality measures.

6.2 Overall Approach

Our approach to analysis of PM strategies is subdivided into three steps:

- i. combine job and data centre *characteristics* to form an overall model;
- ii. structurally describe an advanced PM strategy using a *language*;
- iii. *evaluate* PM strategy for relevant power and performance metrics to assess the strategy qualities.

For the first step of the approach (i), the power and performance models for the data centre simulation framework DACSIM, as proposed in Chapter 4, are configured with advanced PM that allows for control of power states of servers strategically using various observable quantities. In the second step of the analysis approach (ii), the proposed specification of advanced PM strategies from Chapter 5 allows us to structurally describe PM strategies and use a wide variety of state information variables within this simulation framework. The

final step of this approach (iii) is in the remainder of the chapter. The parameters of the power and performance models applied throughout this chapter are described in Section 6.2.1. For illustrative purposes, Section 6.2.2 describes an example strategy using the specification and serves as a running example in the section that follows.

6.2.1 Chapter-Wide Job and Data Centre Configuration

The parameters for the power and performance models that apply to the entire chapter are provided below. The remaining job and data centre configuration are provided in the later parts of this chapter.

The power model for switching power states in a server (cf. Section 4.3.3) is configured with the parameters from Table 6.1 in this chapter.

Figure 6.1 shows the dispatcher that is used in the framework to schedule all the jobs to one of the M servers using *Shortest Queue Next* (SQN). Each server comprises a $G|G|1|\infty|\infty$ queue with a FIFO buffer. As a consequence of PM, the number M of servers available for handling jobs varies over time.

Parameter	R(pc)	R(id)	R(as)	R(of)	R(sl)	R(wk)	R(bt)	R(su)
Assigned Value	200 W	140 W	14 W	0 W	200 W	200 W	200 W	200 W

Table 6.1: Data centre configuration **power** parameters in this chapter.

Figure 6.1: A dispatcher schedules jobs to the queues of the servers 1 to M using *Shortest Queue Next* (SQN).

6.2.2 Example Strategy

For illustrative purposes, we give here an example PM strategy, denoted Θ^{que} , that ensures a good power and performance with the aid of a threshold q by waking and sleeping servers based on queue size (QS) observations. Use of this queue size threshold aims to reduce the (expected) waiting time, and thus the overall (expected) response time. For comparison purposes, the queue sizes threshold is varied between 100 and 1500, with stepsize 100. The satisfier formulas Φ_S^{que} for this queue size threshold strategy with usable global power states $G^{que} = (as, on)$ is as follows:

$$\Phi_S^{que} = \left(\begin{array}{ll} \phi_S^{as} & := \quad (QS \leq q) \\ \phi_S^{on} & := \quad (QS > q) \end{array} \right), \quad (6.1)$$

where $q \in \{100 \cdot i \mid 1 \leq i \leq 15, i \in \mathbb{N}\}$.

This strategy has server constraints $\Phi_C^{que}(s)$ that only allows servers to sleep when the queue of that server has no jobs and servers are only woken when these are actually in power state (PS) asleep, as follows:

$$\Phi_C^{que}(s) = \left(\begin{array}{ll} \phi_C^{as}(s) & := \quad (QS(s) = 0) \\ \phi_C^{on}(s) & := \quad (PS(s) = as) \end{array} \right). \quad (6.2)$$

A recurrence time r of 5.0 s is set for this PM strategy. Smaller values for the recurrence time increase the overhead; higher values of the recurrence time would make the strategy less responsive.

6.3 Strategy Qualities

We assess PM strategies using a combination of four qualities, namely: (i) *efficiency*, (ii) *stability*, (iii) *robustness*, and (iv) *adaptability*. Since every data centre has different clients and environment, the importance of their four qualities might differ. In such a case, decision analysis techniques could assist with finding the best strategy. The meaning of each quality and its quantitative interpretation are elaborated below. To ease the explanation, the strategy from Section 6.2.2 serves as a running example. The four qualities are discussed in the subsequent subsections 6.3.1-6.3.4, followed by a discussion in Section 6.3.5.

6.3.1 Efficiency

Efficiency in PM strategies addresses how well performance and energy goals are met. The goal of performance management is to obtain the lowest response time possible, while the goal of PM is to have the lowest power consumption possible. Since both values are relevant, efficiency is often expressed as the *performance per Watt* (PPW). However, many approaches like [11] struggle with expressing a combination of power consumption and performance in a meaningful way. Since in some cases a power and performance trade-off exists, as already seen in Chapter 3, both power and performance are indicated.

To illustrate the power-performance trade-off, the effect of varying the queue size threshold (as seen in Section 6.2.2) is shown in Figure 6.2. Each dot repres-

Figure 6.2: Varying threshold of the queue (q) that illustrates (i) an efficiency frontier, and (ii) the effect of minor variations in workload and data centre characteristics.

ents a single simulation run with a different queue size threshold. The mean power consumption (on the x -axis) ranges between 2 279 W and 3 053 W and the response time (on the y -axis) ranges between 6.54 s and 78.44 s. The scatter plot shows that by growing or shrinking the queue size threshold performance can be traded for power consumption.

Note that in the same figure an efficiency frontier is drawn that illustrates in which direction the optimal values of the power-performance are found. An efficient advanced PM strategy is considered to have both low mean power consumption and low mean response times. Other details of this figure will be elaborated upon with the other qualities. The shape and position of the curve of the efficiency frontier drawn is based on our analysis with many more variations which results are elaborated in Section 9.5.

An indication of efficiency is the *overhead ratio* (OR), i.e., the mean power-state utilisation of the idle (PU(id)), booting (PU(bt)), waking (PU(wk)), suspending (PU(su)) and sleeping (PU(sl)) ‘overhead’ power states divided by the total power-state utilisation ($\sum_i \text{PU}(i)$). In an efficient PM strategy, the time spent in those ‘overhead’ power states is minimised, because power states should only be switched when really necessary. We can express the OR, as follows:

$$\text{OR} = \frac{\text{PU}(\text{id}) + \text{PU}(\text{bt}) + \text{PU}(\text{su}) + \text{PU}(\text{wk}) + \text{PU}(\text{sl})}{\sum_i \text{PU}(i)}. \quad (6.3)$$

6.3.2 Stability

A *stable* PM strategy ensures acceptable power consumption and performance that does not fluctuate too much as a consequence of incorrect switching between power states of servers. As a consequence of stability, data centres eventually benefit, since less peaks are observed in power consumption, which leads to lower power consumption capacity demands. Customers of data centres are often ensured to have service of good quality via their SLAs, a certain maximum response time threshold below which, for instance, 95% of all response times observed should lie. This demand implicitly requires stable performance.

Recall that the running example (as seen in Section 6.2.2) varies queue size thresholds to obtain a power-performance trade-off. To actually reach the best efficiency for each simulation run, a stable PM strategy is required. A stable PM strategy makes it possible to move each simulation run (a dot) closer to the efficiency frontier (as illustrated in Figure 6.2). The main reason that dots are moving closer to the efficiency frontier is that only necessary power state

switching occurs. As a consequence, the time spent in the ‘overhead’ power states is reduced, which thus improves efficiency.

A convenient method used to meet these SLA demands involves a response time threshold. Counting the number of violations and dividing this number over the total number of samples, estimates the *percentage of jobs that violate the SLA* (SLAv). This method is used in practice by one of our partners. Better PM strategies have a low number of SLA violations.

Another valuable measure is *power state switching frequency* (PSSF), i.e., the number of power state switches per unit time. The PSSF is often the main cause of strong oscillations, because switching between power states leads to changes in the power consumption and performance. One ‘power state switch’ is recorded as soon as a service unit reaches the power state on, as or of (as seen in Figure 4.4). The PSSF is then determined by the *number of power state switches* (#PSS) as a fraction of the *time* (t) elapsed in the entire simulation. We can express the PSSF, as follows:

$$\text{PSSF} = \frac{\#PSS}{t}. \quad (6.4)$$

6.3.3 Robustness

A PM strategy is considered to be *robust* if it is capable of having acceptable stable and efficient performance under minor variations of its data centre configuration. Robustness is relevant for a PM strategy to be applicable under realistic circumstances. Workload often fluctuates during the day and service times of resources vary. These variations include changes in inter-arrival rates λ , service times $1/\mu$ and power state switching time-outs α . So, first a set of relevant minor variants should be formulated. For this, we take the original values, and allow addition or subtraction of 10% of λ , $1/\mu$ and α to their original values. With less than 10% the variants would be too minor to be significant, while with more than 10% would make it harder to compare. Next, stability and efficiency are then compared to the original configuration. In the comparison, the difference in PSSF ($\Delta\text{PSSF} = | \text{PSSF}_{\text{original}} - \text{PSSF}_{\text{variant}} |$) and OR ($\Delta\text{OR} = | \text{OR}_{\text{original}} - \text{OR}_{\text{variant}} |$) values gives a useful indication of robustness. Note that, observation of differences between solely the mean values is considered not to be a correct indication for robustness of the PM strategy, e.g., changing the number of servers will impact the performance per Watt. Therefore, PSSF and OR are more configuration-independent metrics.

In terms of the running example and power-performance trade-off graph (cf. Figure 6.2), a robust PM strategy with minor variations shifts the data

points closer to or away from the efficiency frontier. Note that for variations with regard to the number of servers, a new efficiency frontier is established. Therefore, more generic metrics are required than mean power consumption and mean response times. This is where PSSF and OR give a good indication by comparison with the original model parameters.

6.3.4 Adaptability

When changing from one configuration to another there is a period of adaptation. A PM strategy is considered to be *adaptable* if it adapts itself stably and fast to a change in circumstances. So, an adaptable strategy adds configuration settings to its parameters to be able to determine the right number of servers and the right moment to adapt. The quality adaptability is relevant to consider, because an adaptable strategy is more generally applicable.

Figure 6.3 and Figure 6.4 show the effect of workload fluctuations on the PSSF over small blocks of time ($\Delta t = 20$ s) during a simulation run¹. Both plots display the PSSF over Δt (in h^{-1}) on the y -axis and simulation time (in s) on the x -axis. We observe from comparing the two plots, that PSSF over Δt is temporally much higher with workload fluctuation. The reason for this behaviour is that the employed PM strategy tries to only switch power states when necessary.

Figure 6.5 illustrates a system that enters ($t = 80$ s) and stabilises after an adaptation period ($t = 170$ s), and shows the PSSF over small blocks of time ($\Delta t = 20$ s). The adaptation period is entered by a change in one of the configuration settings (as indicated in the plot). As can be seen in the plot, during the adaptation period the PSSF over Δt is higher and has stronger oscillations.

To observe how adaptable the strategy is, first the *total time of adaptation* (TTA) from change of configuration to a stable situation and the PSSF during this period (denoted as PSSFA) is determined. The PSSFA is the number of power state switching during the adaptation periods ($\#PSSA$) as a fraction of the time spent adapting ($\sum_{i=1}^n TTA_i = TTA_1 + TTA_2 + \dots + TTA_n$). The PSSF is the total number of power state switches during the entire simulation excluding the switches made during adaptation as a fraction of the time spent in a stabilised system ($\sum_1^n t_i = t_1 + t_2 + \dots + t_n$). We now express the PSSFA and a

¹Using job and data centre characteristics from Section 6.4.1 and ADVANCED PM strategy from Section 5.4.3.

Figure 6.3: A PSSF over Δt of a simulation run without workload fluctuations other than the usual stochastic fluctuations in the workload (as mentioned in Section 6.2.1).

Figure 6.4: A PSSF over Δt of a simulation run with workload fluctuation caused by a 30% decrease of the arrival rate.

Figure 6.5: An example adaptation period with PSSF over Δt .

more detailed PSSF, as follows:

$$\text{PSSFA} = \frac{\#\text{PSSA}}{\sum_{i=1}^n \text{TTA}_i}, \text{PSSF} = \frac{\#\text{PSS} - \#\text{PSSA}}{\sum_{i=1}^n t_i}. \quad (6.5)$$

To compute the TTA and PSSFA, a start and an end of the adaptation period has to be determined. The start of this period is easy to detect, since parameters change. The end of this period is determined by observing whether the situation is again stable and efficient as before. Therefore, the strategy has to stabilise to at least the PSSF that it had before the adaptation period started. This requires the strategy to minimally being capable of stabilising.

6.3.5 Discussion

An overview of all computable values for each of the qualities is provided in Table 6.2 based on the previous Section 6.3.2-6.3.4. Also worth noting is that lower values are considered better for all these values.

In the literature [78], the notions of OR and PSSF are related to elasticity in resource management of cloud systems. Their report states that elastic adaptation cannot be described with the traditional performance metrics (response times and utilisation). As a consequence, they present a new set of metrics that

Quality	Computable values	Abbreviation
Efficiency	Mean values	E
	Overhead ratio	OR
Stability	SLA violations percentage	SLAv
	Power state switching frequency	PSSF
	Standard deviation	σ
Robustness	OR difference	ΔOR
	PSSF difference	ΔPSSF
Adaptability	Total time of adaptation	TTA
	PSSF of adaptation	PSSFA

Table 6.2: Observed values for each PM strategy quality (lower is better).

surpasses current approaches on this subject. Our OR is related to a combination of their *accuracy* and *timeshare* metric. Our PSSF is related to their *jitter* metric. Thus, both approaches give expression to similar observations.

An aspect left out of the scope of the quality evaluation is the notion of complexity from PM strategy necessities, which is still open for research. The satisfiers and constraints used for the PM (cf. Section 6.2) require additional sensors, extra computation and/or adjusted infrastructure. This introduces additional overhead by storing additional data, that requires extra space, and additional processing for sensing, computing or storing data, that require extra time and energy. Especially computing satisfiers with hysteresis adds space complexity, since this requires to keep track of information in the past. Moreover, the sampling frequency of observable quantities also requires additional storage and are more computational intensive.

6.4 Evaluation Example

To illustrate the full evaluation of the qualities of a PM strategy, a data centre configuration and its job characteristics are described in Section 6.4.1 as the first step of the three-step approach from Section 6.2. The five literature-inspired strategies AlwaysOn, Optimal, Demotion, Strong and Advanced from Section 5.4 are used for comparison in the example evaluation study. In Section 6.4.2, the last step of the approach evaluates the quality of these five strategies for the given data centre and job characteristics.

6.4.1 Job and Data Centre Characteristics

We consider a data centre in which jobs arrive according to a Poisson process such that the inter-arrival times distribution is exponential with rate λ (job/s). The service time ($1/\mu$ s) of each job is exponential, i.e., the server finishes jobs in a varying amount of time, because of varying job sizes. By default, λ is set to 20.0 job/s and μ is set to 1.0 job/s. Otherwise, the overruling values are stated explicitly.

The daily server utilisation of a typical business data centre based on data from [56] is used (as shown earlier in Figure 5.5). Since the processing speed of these servers remains the same, this server processing utilisation could be rewritten to a time-dependent arrival process with a job arrival rate λ for all servers of $\frac{(\mu \cdot n)}{100} \cdot \text{PU}(\text{pc}, \text{ins})$, where $\text{PU}(\text{pc}, \text{ins})$ is the currently observed processing utilisation (in %) and $(\mu \cdot n)$ is the maximum allowable number of jobs

arriving per second. This assumes that overhead caused by scheduling jobs is negligible.

The advantage of this approach is that data centres are simulated by only analysing its day-night patterns. This can be done by deriving the inter-arrival times of jobs for each hour together with its job characteristics such as demands, priorities and job sizes. Validation and sensitivity analysis of such parameters is considered to be future work.

Job scheduling is an essential part of the data centre configuration, since it influences the performance a lot. Jobs arrive at the data centre and are distributed via a dispatcher. The dispatcher uses the default scheduling of jobs via *shortest queue next*. However, jobs can only be scheduled to servers in the idle and processing power states.

The model for power state switching supports the three global power states, as in Figure 4.4. Each state in the model has a specific power consumption. By default, the time required for a server to shut down α_{su} is set to 100.0 s, boot α_{bt} is set to 100.0 s, wake α_{wk} is set to 10.0 s and sleep α_{sl} is set to 10.0 s and each time is deterministic. The awareness of power states in the models allows us to compute power consumption (P) by rewarding each power state with a power consumption (as assigned in Table 6.1). Note that Processing is the only power state in which jobs are served.

The mean power consumption ($E[P]$) and mean response times ($E[R]$) are computed using the batch means method. This method requires the model time (t_{sim}) to be very long, which is usually around 86 400 virtual seconds (exactly 1 day/night cycle), and the system should finish its start-up phase after some warm-up (*wup*) period. The maximum number of servers available is 30. The Server-Level Agreements (SLAs) demand a response time threshold (R_{SLA}) set to 20s. Violations of this threshold could result in a penalty or are considered to be unacceptable. The window size for the (exponentially) moving averages are set to one second of sample data. Analysis of different window sizes is part of future work.

6.4.2 Quality Evaluation

A global overview of an assessment of the above PM strategies is provided in Table 6.3. Each row in the table represents a PM strategy and each column a quality. In each cell, several measurements for that specific strategy and quality combination are provided. Beside the measurements itself, each measurement is ranked by comparing it to the other strategies, where the highest ranks (lowest numbers are best) indicate the most optimal values. We combine measure-

Strategy	Efficiency	Stability	Robustness	Adaptability
AlwaysOn (Θ^{all})	(11) $= \sum_{ranks}$	(5)	(6)	(2)
	(5) $E[E] = 115.10 \text{ kWh}$	(2) $\sigma_{(P)} = 163.910 \text{ s}$	(5) $\text{avg.}\Delta\text{OR} = 1.38 \%$	(1) $\text{avg. PSSFA} = 0 \text{ h}^{-1}$
	$E[P] = 4796 \text{ W}$	(1) $\sigma_{(R)} = 0.328 \text{ s}$	(1) $\text{avg.}\Delta\text{PSSF} = 0 \text{ h}^{-1}$	(1) $\text{avg. TTA} = 0 \text{ s}$
	(1) $E[R] = 1.269 \text{ s}$	(1) $\text{SLAV} = 0 \%$		
	(5) $\text{OR} = 66.86 \%$	(1) $\text{PSSF} = 0 \text{ h}^{-1}$		
Optimal (Θ^{opt})	(12)	(20)	(6)	(10)
	(4) $E[E] = 59.09 \text{ kWh}$	(5) $\sigma_{(P)} = 494.238 \text{ s}$	(2) $\text{avg.}\Delta\text{OR} = 0.21 \%$	(5) $\text{avg. PSSFA} = 331.2 \text{ h}^{-1}$
	$E[P] = 2462 \text{ W}$	(5) $\sigma_{(R)} = 5.590 \text{ s}$	(4) $\text{avg.}\Delta\text{PSSF} = 6.12 \text{ h}^{-1}$	(5) $\text{avg. TTA} = 150.0 \text{ s}$
	(5) $E[R] = 8.761 \text{ s}$	(5) $\text{SLAV} = 2.57 \%$		
	(3) $\text{OR} = 4.69 \%$	(5) $\text{PSSF} = 241.2 \text{ h}^{-1}$		
Demotion (Θ^{dem})	(10)	(16)	(8)	(8)
	(2) $E[E] = 55.06 \text{ kWh}$	(4) $\sigma_{(P)} = 487.528 \text{ s}$	(3) $\text{avg.}\Delta\text{OR} = 0.24 \%$	(4) $\text{avg. PSSFA} = 293.8 \text{ h}^{-1}$
	$E[P] = 2294 \text{ W}$	(4) $\sigma_{(R)} = 5.463 \text{ s}$	(5) $\text{avg.}\Delta\text{PSSF} = 7.56 \text{ h}^{-1}$	(4) $\text{avg. TTA} = 145.83 \text{ s}$
	(4) $E[R] = 8.665 \text{ s}$	(4) $\text{SLAV} = 1.879 \%$		
	(4) $\text{OR} = 4.78 \%$	(4) $\text{PSSF} = 237.6 \text{ h}^{-1}$		
Strong (Θ^{str})	(6)	(8)	(3)	(6)
	(3) $E[E] = 56.14 \text{ kWh}$	(1) $\sigma_{(P)} = 147.939 \text{ s}$	(1) $\text{avg.}\Delta\text{OR} = 0.16 \%$	(3) $\text{avg. PSSFA} = 51.48 \text{ h}^{-1}$
	$E[P] = 2339 \text{ W}$	(3) $\sigma_{(R)} = 3.393 \text{ s}$	(2) $\text{avg.}\Delta\text{PSSF} = 2.52 \text{ h}^{-1}$	(3) $\text{avg. TTA} = 110.42 \text{ s}$
	(2) $E[R] = 4.257 \text{ s}$	(2) $\text{SLAV} = 0.0021 \%$		
	(1) $\text{OR} = 3.59 \%$	(2) $\text{PSSF} = 9 \text{ h}^{-1}$		
Advanced (Θ^{adv})	(5)	(11)	(7)	(4)
	(1) $E[E] = 53.02 \text{ kWh}$	(3) $\sigma_{(P)} = 170.192 \text{ s}$	(4) $\text{avg.}\Delta\text{OR} = 0.31 \%$	(2) $\text{avg. PSSFA} = 33.84 \text{ h}^{-1}$
	$E[P] = 2209 \text{ W}$	(2) $\sigma_{(R)} = 3.360 \text{ s}$	(3) $\text{avg.}\Delta\text{PSSF} = 3.24 \text{ h}^{-1}$	(2) $\text{avg. TTA} = 106.25 \text{ s}$
	(2) $E[R] = 4.101 \text{ s}$	(3) $\text{SLAV} = 0.0130 \%$		
	(2) $\text{OR} = 3.90 \%$	(3) $\text{PSSF} = 14.4 \text{ h}^{-1}$		

Table 6.3: PM strategies qualities assessment (cf. Table 6.2).

ments by giving equal weight to each relevant computed value and average its rank for each of the qualities. In practice, these weights could be adjusted if some values are considered to be unacceptable.

A relevant observation from Table 6.3 is the 54% energy reduction for the Advanced strategy (53.02 kWh for 1 day) compared to using the AlwaysOn strategy (115.10 kWh for 1 day). However, such an energy reduction caused by PM has effect on other quantities of the data centre. The Optimal and Demotion strategies both show already a large improvement in energy efficiency with reasonable performance, stability, robustness and adaptability. With additional fine tuning, the Strong strategy adds even more stability and robustness, and a boost in performance. The Advanced strategy shows even less energy consumption with still fine efficiency, stability, slightly improved adaptability and slightly less robustness compared to the Strong strategy.

All these observations together show that the right strategy depends on the demands of this data centre on each of the qualities. Some data centres find certain percentage of SLA violations unacceptable. Other data centres might have very steady workload characteristics, which makes robustness and adaptability less relevant. Therefore, these two examples might choose different PM strategies based on their quality demands.

6.5 Conclusions

For the purpose of analysing energy efficiency and performance in data centres, this paper introduces novel metrics for evaluation of PM strategies in four qualities: (i) efficiency, (ii) stability, (iii) robustness and (iv) adaptability.

First, the job and data centre characteristics have been described. Subsequently, five PM strategies have been specified to meet data centre quality demands for some global and server-level conditions with the aid of so-called satisfiers and constraints formulas. In the final step of the approach, these PM strategies have been evaluated with our novel metrics.

The various qualities are assessed for a data centre configuration with 30 servers and typical small business data centre workload. An energy reduction of 54% is obtained by PM strategies (compared to AlwaysOn strategy) inspired by the literature (Optimal and Demotion) and fine tuning thereof (Strong and Advanced). For these fine tuned strategies, energy efficiency is increased *and* performance, stability, adaptability and robustness are maintained as well.

Integrated Model

Advanced power management (PM) and cooling techniques for data centres often co-exist as separate entities in current-day operation of data centres. This chapter proposes to combine these techniques to achieve greater power savings. To this end, an existing thermal-aware model is integrated in an extensive simulation framework for data centres using power and performance models, which allows for a detailed study in power, performance and thermal metrics. The chapter compares four distinct cases for studying the effect on these metrics: a data centre with (i) basic functionality; (ii) advanced cooling; (iii) advanced PM; and (iv) a combination thereof. The combined case shows a reduction in the energy consumption compared to the other cases while performance and thermal demands are kept intact. The combination of these techniques shows improvements in energy savings and shows it is meaningful to investigate further into smart combined energy saving techniques.

The chapter is organised, as follows. First, an introduction is provided in Section 7.1. In Section 7.2 the models are introduced and their integration into the simulation framework is discussed. Next, the simulation configuration is given in Section 7.3 and different control scenarios are described in Section 7.4. Finally, the simulation results are studied in Section 7.5. Conclusions are provided in Section 7.6.

This chapter is based on the publication [B3] that has been performed in collaboration with research project partners from RIJKSUNIVERSITEIT GRONINGEN.

7.1 Introduction

Recall that the main energy-efficiency improvements described in Section 2.1.3 are (i) advanced cooling strategies, (ii) power proportionality through PM and load balancing and (iii) server consolidation through virtualisation. The advanced cooling strategies focus on techniques that increase the thermal efficiency of the data centre like hot aisle isolation, economizers and liquid cooling. Power proportionality is achieved with PM software and hardware and load balancing techniques, whereas server consolidation focuses on running current workload on as few servers as possible, in order to decrease the amount of hardware necessary in the data centre.

While these three areas separately show many improvements, we believe that more improvements can be gained by combining these areas, specifically combining the area of advanced cooling strategies with the area of power proportionality through PM and job scheduling. In this chapter, we investigate the cooperation between strategic PM control and strategic thermal control. Besides possible energy consumption benefits, this study allows us to show the general applicability of both these modelling approaches.

In Chapter 4–6, a simulation framework has been introduced to analyse models for both power and performance in data centres that use PM techniques to reduce its energy consumption. In this framework it is easy to study power and performance metrics of high-level models for any given data centre configuration and workload characteristic. Already these kind of analyses provide helpful insights in the design phase of data centres. Simultaneously a theoretical thermodynamical characterization of the cooling system in data centres has been performed [181]. It proposes models for thermal-aware data centres that provide insights in the thermodynamics of the air flows in data centres. A control strategy was developed to dynamically steer the data centre to the optimal job distribution and cooling temperature to achieve the lowest energy consumption of the cooling system.

In this chapter, we focus on integrating the thermodynamical model in the existing simulation framework to study the interaction between the PM strategies from Chapter 4–6 and the thermal-aware controllers from [181]. Individually these areas have received much attention by researchers, e.g., [73] and references therein, however the combination of these two fields is much less studied [193]. This chapter contributes to the existing state-of-the-art by providing an extensive simulation study that shows the viability of combining these two distinct control strategies and study the improvements that can be made by combining the two approaches.

7.2 Model Integration

In this section we explain how the different models are integrated in the simulation framework as well as explaining the background of each of the models.

Figure 7.1 shows an overview of how our models from *University of Twente* (UT) are connected to the models from *Rijksuniversiteit Groningen* (RUG) and how they interact with each other. This overview shows that the models use characteristics from the data centre from an *Industrial Partner* (IP), and computed results from the other models as input. All models use job and data centre characteristics based on realistic data centre configurations, (1)–(3). The temperature of each unit, (4a), is communicated from the thermodynamical models to the power-performance models for analysis or for use in PM related decisions; the temperature of each unit is computed with the power consumed by the units, (4b). The energy consumption of the computer room air conditioning (CRAC) is calculated using the thermodynamics and is communicated to the cascade model, (8); the energy consumption of the other infrastructural components remain linearly dependent on the energy consumed by IT equipment, (5). The total data centre energy consumption is sent to the advanced PM module, (10), where strategic PM related decisions are made, (7a), based on the metrics available in the models, (7b). Advanced cooling control decisions are made with the computed constants from the thermodynamical models and unit temperatures, (6). The optimal (from thermal-aware perspective) job distribution and supply temperature, (9), are communicated to the power-performance models for analysis or, if thermal-aware job dispatching is enabled, the dispatcher uses the optimal distribution to distribute jobs among the servers.

The communication frequency between the models and data centre, is as follows. The models are required to be calibrated with the characteristics from the data centre, (1)–(3), only once at the start of the simulation. Since the thermal-aware models depend on the power consumed of the units, (4b), we need to update unit temperatures, (4a), and power consumption of the cooling, (8), every time the power consumption of a rack changes. We simplified this for the ease of computation, such that the simulation framework updates its temperatures metrics with a relatively high frequency of 0.1 s. Each time a job arrives in the data centre and jobs are scheduled in a thermal-aware fashion, the optimal job distribution is computed and communicated to the job dispatcher, (9), which decides where the job should go based on an optimal distribution.

First, the hierarchy of the data centre infrastructure is described in Section 7.2.1. The thermodynamical models for such a hierarchical system are

Figure 7.1: Detailed integration of thermal-, power- and performance-aware models for data centres.

Figure 7.2: Visualisation of the organisational hierarchy in data centres.

provided in Section 7.2.2. The power and performance models are extended to support the hierarchical system in Section 7.2.3. Advanced cooling control that adapts the supply temperature of the CRAC unit and desired job distribution are elaborated in Section 7.2.4. Section 7.2.5 elaborates the PM strategies used in this chapter that follow the specification from Section 5. The integration within the simulation framework DACSIM is elaborated in Section 7.2.6.

7.2.1 Hierarchical Infrastructure

Data centres can be represented as a hierarchical system, see Figure 7.2. At the highest level, the *rack* level, data centres typically consist of aisles of racks. These racks typically consist of multiple subunits each containing a multitude of servers. This is the second level of the hierarchy, the *unit* level. On the lowest level, the *server* level, the servers consist of multiple computing *cores*.

7.2.2 Thermodynamical model

The aisles in a data centre are typically configured in cold and hot aisles. The cold air from the CRAC enters the data centre via the cold aisles and goes through the racks where it extracts some of the heat which is produced in the racks. The heated air then exits the racks into the hot aisles from where it is led to the CRAC to be cooled down to the desired cooling temperature. Since there are inefficiencies in the system, not all of the exhaust air is returned to the CRAC, and some of it is recirculated back to the cold aisles. There it mixes with the supplied air from the CRAC, heating the cool air slightly and reducing the effectivity of the CRAC. This inefficiency is the basis of the thermal models in [181]. This section briefly restates the models but leaves the details to the cited work. The models presented below model the temperature dynamics in the data centre and relate the temperature of the units to the energy required to cool the data centre.

Power consumption of units. The power consumption of the units in the racks is modelled with a linear function, i.e., it consists of a load-dependent and load-independent part. The load-independent part depends on the current global power state of the servers in the unit, which is further elaborated on in Section 7.2.3. The load-dependent part depends on the number of processors that are actively processing jobs. The power consumption of the units is

$$P(t) = V + WD(t), \quad (7.1)$$

where

$$\begin{aligned} P(t) &= \begin{pmatrix} P_1(t) & P_2(t) & \cdots & P_n(t) \end{pmatrix}^T, \\ V &= \begin{pmatrix} v_1 & v_2 & \cdots & v_n \end{pmatrix}^T, \\ W &= \text{diag}\{w_1, w_2, \cdots, w_n\}, \\ D(t) &= \begin{pmatrix} D_1(t) & D_2(t) & \cdots & D_n(t) \end{pmatrix}^T. \end{aligned}$$

Here, $P_i(t)$ is the power consumption of unit i at time t , v_i [Watts] is the power consumption of unit i related to the current power state of its servers, w_i [Watts CPU⁻¹] is the power consumption per CPU in use in unit i , and $D_i(t)$ the number of CPU's that are actively processing work in unit i at time t .

Thermodynamics. The temperature change of each unit in the data centre can be expressed by a differential equation which captures the relation between current temperature, supply temperature and power consumption of the units

$$\frac{d}{dt}T_{\text{out}}(t) = A(T_{\text{out}}(t) - \mathbb{1}T_{\text{sup}}(t)) + M^{-1}P(t). \quad (7.2)$$

Here

$$T_{\text{out}}(t) = \begin{pmatrix} T_{\text{out}}^1(t) & T_{\text{out}}^2(t) & \cdots & T_{\text{out}}^n(t) \end{pmatrix}^T,$$

and T_{out}^i [°C] is the temperature of the exhaust air at unit i , and T_{sup} [°C] is the temperature of the cool air supplied by the CRAC. Furthermore

$$\begin{aligned} A &= \rho c_p M^{-1}(\Gamma^T - I_n)F, \\ F &= \text{diag}\{f_1, f_2, \cdots, f_n\}, \\ M &= \text{diag}\{c_p m_1, c_p m_2, \cdots, c_p m_n\}, \\ \Gamma &= [\gamma_{ij}]_{n \times n}. \end{aligned}$$

Here, ρ [kg m⁻³] is the density of the air, c_p [J °C⁻¹ kg⁻¹] is the specific heat capacity of air, m_i [kg] is the mass of the air inside the unit, f_i [m³ s⁻¹] is the velocity of the air flow through unit i , and Γ is the matrix containing all the recirculation parameters.

Power consumption of CRAC. The power consumption of the CRAC is determined by the amount of energy which has to be extracted from the air. This in turn depends on the temperature of the air flows in the data centre,

and the efficiency factor of the CRAC unit. The amount of energy to be extracted, $Q_{\text{rem}}(t)$, can be written in terms defined in the thermodynamical model. The efficiency of the CRAC is determined using the coefficient of performance, $\text{COP}(T_{\text{sup}}(t))$, first defined by Moore et al. [128]. The COP is a function of the target supply temperature and is defined such that a higher value denotes a more efficient CRAC unit. The heat removed and the corresponding CRAC energy consumption is given by

$$Q_{\text{rem}}(t) = -\mathbb{1}^T M A(T_{\text{out}}(t) - \mathbb{1}T_{\text{sup}}(t)), \quad (7.3)$$

$$P_{AC}(T_{\text{out}}(t), T_{\text{sup}}(t)) = \frac{Q_{\text{rem}}(t)}{\text{COP}(T_{\text{sup}}(t))}. \quad (7.4)$$

In [128] the COP for a water-chilled CRAC unit in the HP Utility Data Center is characterized as a quadratic, increasing function in the range of operation for T_{sup} .

7.2.3 Power and Performance Models

The models for power and performance are based on earlier work from Chapter 4. Here we explain how each model is adapted to fit in the framework of this chapter.

Performance. The performance models are extended with a two-level scheduling algorithm, see Figure 7.3. A central dispatcher distributes jobs to one of the n units using a scheduling algorithm of choice. Then, jobs are scheduled in round-robin fashion to servers 1 to N inside the unit. As in the original work, each server comprises a $G|G|1|\infty|\infty$ queue with a FIFO buffer.

Since an optimal job distribution is proposed to the job dispatcher, and jobs are *not* allowed to migrate instantaneously from one server to another, an interpretation of the optimal job distribution has to be made within the job dispatcher for newly arriving jobs that have to be allocated. A good solution should allocate newly arriving jobs to units that make progression towards the optimal job distribution. So, when a new job arrives in the data centre, the unit, with the largest difference from the optimal job distribution, receives the newly arrived job.

Power consumption of IT equipment. The power consumption at time t for each of these servers is equal to the predefined reward $R(k)$ for each power state k as shown earlier in Figure 4.4. Each state has a fixed power consumption with the exception of the processing state. As each server can have multiple computing cores, the power consumption of the processing state is also

Figure 7.3: Our extended dispatcher schedules jobs to the queues of the servers 1 to N via the hierarchy of the units 1 to n using a two-level scheduling algorithms in *Thermal-Aware* (TA) and *Round-Robin* (RR) fashion.

dependent on the number of active cores. The reward for the processing state is therefore given by $R(\text{pc}) = R(\text{id}) + w_i D_s(t)$, where D_s is the number of active cores in server s , and w_i is, as in Section 7.2.2, the power consumption per active core for the unit the server resides in. The power consumption of unit i is then given by the sum of the power consumption of the servers inside the unit.

The main PM feature is the ability to switch between global power states. This allows to adapt power consumption levels at the cost of time spent switching between global power states and therefore decreased performance.

Power consumption of data centre. Based on the IT equipment an estimation of the power consumed by other necessary infrastructural components can be computed using simple linear functions, according to the *cascade model* (as discussed in Section 4.3.4). The total data centre power consumption is calculated by the sum of the power consumption of the CRAC, P_{AC} from (7.4), and the power consumption of the IT equipment and the other infrastructural components.

7.2.4 Advanced Cooling Control

The thermodynamics described in Section 7.2.2 are used by the authors in [181] to characterise an optimal operating point to minimise the power consumption of the cooling system. This optimal operating point involves both an optimal

CRAC supply temperature and an optimal workload distribution. Furthermore, controllers are designed that can dynamically steer the data centre to this optimal operating point. The controllers are given by

$$\frac{d}{dt}T_{\text{sup}}(t) = \mathbb{1}^T A^T Z(T_{\text{out}}(t) - T_{\text{safe}}), \quad (7.5)$$

$$\frac{d}{dt}D(t) = \left(\frac{\mathbb{1}\mathbb{1}^T}{n} - I_n\right)(M^{-1}W)^T Z(T_{\text{out}}(t) - T_{\text{safe}}), \quad (7.6)$$

where T_{safe} could be the equipments' safe temperature level set by the manufacturer or a safe human working condition set by the data centre owner, and Z is a positive definite matrix.

The controller aims at steering the temperature distribution to the safe temperature level. The values of the inputs will be adjusted as long there are deviations from the safe temperature level.

When applying these controllers in the simulation, the values obtained from the controllers are used as set points. The CRAC takes the set point for $T_{\text{sup}}(t)$ and steers the supply temperature to this value, and the set point for $D(t)$ is sent to the job dispatcher each time a new job arrives. The job dispatcher then compares the actual job distribution to the given set point for $D(t)$ and assigns the job to the unit which shows the largest deviation from the desired distribution. The controllers work only for a specified range of workloads. The exact range depends on the values of the parameters of the data centre, e.g., power consumption of servers, and recirculation flow. For our set of parameters the controllers work up to workloads of 50% of the total data centre computing capacity. The above controllers are therefore disabled for higher workload levels. The nature of this restriction is explained further in [181].

7.2.5 Advanced Power Management

An advanced PM strategy strategically puts servers in lower global power states, e.g. the low power consumption *Asleep* state and *Off* power state, to reduce overall energy consumption while performance is kept intact. The power consumption for each power state is depicted in Figure 4.4 with rewards R that are server configuration specific. Each r seconds a predefined PM strategy determines if servers should switch global power states. Following the specification from Chapter 5, a PM strategy, Θ , is defined by the global-level constraints, Φ_S , and the server-level constraints, $\Phi_C(s)$, for each of the available global power states in G . The strategy used is the Advanced strategy as described in Section 5.4.3. The only difference is that fraction of all servers

that are idle before a server is put to sleep or turned off is changed to a larger bound from 20% to 30% in (5.12b) and (5.12d). This provides a safety bound that allows for the thermal-aware distribution of jobs to remain effective.

7.2.6 Integration in DACSIM

For the purpose of integrating the thermal-aware models in DACSIM, the matrix library EJML [1] is included to handle the differential equations. A module is set up that allows for (i) all the computations related to the thermal-aware models, (ii) transient analysis of the computed values during a simulation run and (iii) full logs of all the computed values.

Figure 7.4 shows the low-level design of the thermal module using a UML Class Diagram structure. The simulation framework is extended with a thermal module similar to the PM module shown in Figure 5.6. The ModThermal class contains all the functions computations of the temperature of the racks, computation of the desired job distribution, control of the supply temperature and power consumption of the cooling. The constructor ModThermal initialises the physical parameters v_i , w_i , ρ , c_p , m_i , f_i , γ_{ij} and connects the module to a data centre dc with N racks. The function `initConstants` computes the initial

Figure 7.4: Low-level design UML Class Diagram structure for the thermal module.

values of the constants M , M^{-1} , I_n , A , W , Z , $\mathbb{1}$, T_{safe} based on the physical parameters. These constants are required to be computed only once for the entire simulation at the start. The initial temperatures of the server racks are set with calls of the function `initTemp`. The functions `pullTempPower`, `pullPower` and `pullJobDistr` are used to respectively retrieve the previously observed rack temperatures $T_{\text{out}}(t)$, rack power consumption $P(t)$ and jobs distributed to the rack $D(t)$. These pull functions are used in combination with the constants to compute (i) new temperatures of the racks with the function `computeTemp`, (ii) newly proposed job distribution with the function `controlJobDistr`, (iii) new supply temperatures with the function `controlSupplyTemp`, and (iv) power consumed by the CRAC unit with the function `computePAC`. The functions `getT-Sup` and `getDNew` are used by remote classes to retrieve respectively the supply temperature and the newly proposed job distribution as these values are only stored in the module.

7.3 Model Parameters and Output

In this section, we describe the parameters and output of the simulation models. Therefore, we describe the job and data centre characteristics in Section 7.3.1 and settings specific to the simulation in Section 7.3.2.

7.3.1 Job and Data Centre Characteristics

The data centre in the simulation consists of 30 Dell PowerEdge 1855 server racks, i.e. units in Figure 7.2. Each unit has 10 dual-processor blade servers, i.e. a total of 20 CPU cores per unit. The base power consumption of a server in an idle state is $R(\text{id}) = 172.8$ W. The power consumption of each active CPU core is $w_i = 145.5$ W [165]. The power consumption of server s in the sleep or off power states is respectively $R(\text{as}) = 14$ W and $R(\text{of}) = 0$ W [64]. All other power states $R(\text{wk})$, $R(\text{sl})$, $R(\text{bt})$ and $R(\text{su})$ for global power state switching are rewarded as if all CPUs in the server are in use. The global power state switching time is distributed deterministically with mean $1/\alpha_{\text{wk}} = 1/\alpha_{\text{sl}} = 0.1$ (10 s) and $1/\alpha_{\text{bt}} = 1/\alpha_{\text{su}} = 0.01$ (100 s). The coefficients of the cascade model are taken from Section 4.3.4.

The data centre parameters were obtained from measurements by at the IBM Zurich Research Laboratory [184]. The safe temperature threshold for the units is set at 30 °C. The initial temperature of all units is set to 27.5 °C.

Jobs arriving at the data centre are characterised by HTTP requests. The inter-arrival times and service times distributions in the model are calibrated

with two data sets of HTTP requests from a real data centre, with each set having a duration of about 21 days (about 27.2 million entries), using a fitting algorithm (in cooperation with Better.be). These distributions are exponential with a rate λ that is proportional to the desired workload in the case of the inter-arrival time, and a mixture of normal distributions with an average service time of about 107 ms in the case of the service times. The Service-Level Agreement (SLA) requires response times of HTTP requests to be below 1 s and an average response time of 300 ms.

7.3.2 Simulation Settings

Time units are set to seconds. The duration of the simulation was 3600 s. The warmup period for the system to adapt to the initial transient phase (e.g. sleeping the right number of servers) has been set to 1000 s. All simulations have been performed on a machine equipped with a 2.70 GHz INTEL® CORE™ i7-4800MQ CPU, 8 GB of RAM and WINDOWS 7 64-bit with AnyLogic v8.1.0. The execution time of a single simulation run was between approximately 1 minute for the lowest workload and approximately 30 minutes for the highest workload. Results required a total of 40 simulation runs.

7.4 Case studies

This section focuses on four different control scenarios in a realistic data centre setting for the purpose of studying the impact of each control strategy on energy, performance and thermal measures. In Table 7.1 an overview of the different scenarios is given.

In the **base case** scenario (*Scenario I*), *no* advanced control mechanics are applied, i.e., there is *no* feedback in control decisions. This scenario represents current day heuristics in many data centres. In this scenario, basic control of the data centre is applied at two levels, namely (i) *cooling*, (ii) *job scheduling*. The supply temperature of the cooled air of the CRAC is controlled in a way that keeps the temperature of the units below a certain safe threshold. If the maximum unit temperature is above the safe threshold, then the supply temperature will decrease, otherwise it will increase. Jobs arriving at the data centre are scheduled in round-robin fashion to the units. The PM strategy is inactive, i.e., all servers are always turned on.

In the **advanced cooling strategy** scenario (*Scenario II*), only advanced cooling control is applied, there is *no* active PM, i.e., servers are always turned on.

Scenario	Scheduler	Cooling	PM Strategy
I: Base Case	RR-RR	Static	Always On
II: Advanced Cooling	TA-RR	Dynamic	Always On
III: Advanced PM	RR-RR	Static	Strategy Θ
IV: Combined	TA-RR	Dynamic	Strategy Θ

Table 7.1: Overview of the four scenarios

Controllers (7.5) and (7.6) are applied according to the steps described in Section 7.2.4. This control is tested up to and including a workload of 50% of the total data centre workload capacity.

In the **advanced PM strategy** scenario (*Scenario III*), cooling control and job distribution are the same as in the base case, whereas advanced PM strategy (5.12) and (5.13) is applied as specified in subsection 7.2.5.

The **combined cooling and PM strategy** scenario (*Scenario IV*) allows for the investigation of a combination of both advanced PM strategies and advanced thermal-aware control. In this scenario, global power states are switched according to strategy (5.12) and (5.13) for energy-efficiency, and the job dispatcher follows the set point of the job distribution using controllers (7.5) and (7.6) for thermal-efficiency. Same as in Scenario II, advanced cooling control is applied up to workloads of 50% of the total data centre capacity. For workloads higher than 50%, this scenario operates according to Scenario III.

7.5 Results

In this section, we discuss the effects observed from simulating the four scenarios for different levels of workload. Therefore, we elaborate the three metrics: (i) *energy consumption* (in Section 7.5.1); (ii) *performance* (in Section 7.5.2); and (iii) *thermodynamics* (in Section 7.5.3).

7.5.1 Energy

The total expected energy consumption $E[E]$ of the data centre for the full duration of the simulation is plotted in Figure 7.6 for all the scenarios, with different utilisation levels varying from 0% to 90% with increments of 10%. Note that with a processing utilisation of 100% the system would become unstable.

Figure 7.5: The temperature difference (ΔT) between the average maximum and minimum temperature for the four scenarios with varying workloads from 0% to 90% of the total data centre capacity in increments of 10%.

Figure 7.6: Total expected energy consumed by the data centre for the four scenarios with varying workloads from 0% to 90% of the total data centre capacity in increments of 10%.

Figure 7.7: Mean response time for the four scenarios with varying workloads from 10% to 90% of the total data centre capacity in increments of 10%.

First, it is observed from Figure 7.6 that the higher the utilisation level becomes, the larger the energy reduction of the *advanced cooling* strategy becomes with respect to the *base case*. Secondly, a larger energy reduction is observed at lower utilisation levels when only *advanced PM* is applied. However, the best energy savings, for all utilisation levels, are obtained when the two control approaches are combined as in Scenario IV. At higher utilisation levels, our strategies have almost no room to control anything, and therefore no significant energy savings are observed.

7.5.2 Performance

For each of the simulation runs, the Service-Level-Agreement violations are recorded as a percentage of the overall number of jobs. The percentage of SLA violations for all processing utilisations has been 0% with an outlier of 0.011% at 90% processing utilisation due to the stochastic nature of the simulation.

Figure 7.7 shows the mean response times for the four scenarios with varying workload from 10% to 90%. The 0% case is skipped as there are no jobs arriving in the system in this case. The figure shows an increase of at most 100 ms in the average response times for all scenarios. It is seen that distributing jobs in a thermal-aware fashion gives rise to a slight increase in response

times, with the biggest impact seen at 50% utilisation. The SLA requirements are still met, however, because response times should be at most 1 s and the average response time should not exceed 300 ms. So, the overall performance is largely maintained while energy is being saved.

7.5.3 Thermodynamics

In order to plot the temperature data in an understandable way, the variation in temperatures among the units is studied. To do this, the difference between the average maximum and minimum unit temperature over the full simulation run is calculated for all simulation runs. This temperature difference is shown in Figure 7.5.

Comparing the temperature differences of Scenario I with Scenario II, we see that the *advanced cooling* strategy results in a very balanced temperature profile among the units. This is the reason for the energy savings between the two scenarios, observed in Figure 7.6. When comparing the temperature differences between Scenario III and Scenario IV, we see again large improvements in favour of the *combined case*, where *advanced cooling* is applied. Same as before, this smaller variation results in less energy consumed.

Note that in the case of 0% workload, not much interesting can be done as there are no jobs available for redistribution. Also in the case of 10% workload it is seen that Scenario IV has an increased variation compared to Scenario III. However when considering all units, less heat is generated overall, as can be deduced from the lower energy consumption of Scenario IV in Figure 7.6.

7.6 Conclusions

In order to analyse a potential power-, performance- and thermal-aware data centre, thermodynamical models have successfully been integrated in an existing extensive simulation framework with power and performance models. Moreover, advanced energy-aware control strategies are studied in a realistic simulation setting. Energy consumption, performance and thermodynamics are analysed in four scenarios where different control strategies are applied. From the simulation runs we see that combining thermal-aware control with power- and performance-aware strategies yields the best energy savings without suffering any SLA violations. Furthermore, we showed that the thermal-aware controller successfully balances output temperatures of the units.

Part II

Experimental Validation

Workload Modelling for Model Calibration

The modular simulation model from Chapter 4 has been developed for the purpose of considering both performance and energy in order to study power-performance trade-offs in data centre design. Key to such studies is the availability of a workload model. In this chapter, we present a workload characterisation model and algorithm using modern-day data science techniques, building on top of JUPYTER NOTEBOOK and the PROFIDO platform. We present the method and show its versatility on a case study with real-world traces of 20 million entries, provided by the Dutch company BETTERBE.

This chapter is organised, as follows. Section 8.1 provides an introduction with motivation and goals of this chapter. We first present the overall “data science approach” in Section 8.2, before we discuss the data preparation phase in Section 8.3, followed by the actual algorithm, including background and implementation in Section 8.4. We then experimentally validate the approach in Section 8.5, and conclude the chapter in Section 8.6.

This chapter is based on the publication [B1] that has been performed in collaboration with BETTERBE and a supervised student.

8.1 Introduction

In Chapter 4, a modular simulation model (based on ANYLOGIC) has been developed that can be used to study performance-energy trade-offs in data centre design. Key to such studies is the availability of a workload model.

There is a long history on the characterisation of computer and communication system workloads, dating back more than a century to the seminal work of Erlang, but also to the 1970s when multi-programming systems were being developed, cf. the work of Ferrari [60]. A nice overview of relevant work has been presented by Calzarossa and Serazzi [32]. In the area of communication systems there has been an emphasis on the modelling of correlated arrival stream, such as provided by Markov-modulated arrival processes [61] and self-similar traffic [12], [111]. The availability of good fitting techniques for so-called phase-type distributions, see, e.g., [55], [153] enhanced the potential of fully Markovian analysis.

With this chapter we pursue a slightly different approach. Since we do *not* aim for an analytical or numerical model evaluation, but instead to discrete-event simulations, we do *not* necessarily have to restrict ourselves to phase-type distributions or other types of models that fit more classical performance evaluation techniques. Instead, when investigating workloads for data centres, we found the workloads to be often of “mixed normal” distribution form [115]. This observation is also intuitively appealing: a data centre typically supports a wide variety of applications, with very different service times. The combined workload model is then a mixture of variations around a number of (different) means. A mixture of exponentials often has difficulty to describe such behaviour, due to the support starting from zero onwards.

For that reason, we present a fully automated algorithm, built using JUPYTER NOTEBOOK [149] in combination with the PROFIDO platform [18] that, once fed with an appropriate measurement trace, computes an appropriate mixed normal distribution fully automatically. We present the method and show its versatility on a case study with real-world traces of 20 million entries, provided by the Dutch company BETTERBE.

8.2 Data Science Approach

According to [157], data science can be seen as a process and be applied for meaningful analysis (cf. Figure 8.1). First, raw data sets are collected (2) from the data centres (1). These raw data sets contain relevant information about

Figure 8.1: Applied data science process (based on [157]).

the jobs. These raw data sets need to be processed (3) with the aid of a tool chain for data science. Before the data sets are ready to be analysed, the data sets need to be cleaned (4) from irrelevant data and missing values. Exploratory data analysis (5) is part of preliminary research, that helps to collect better data and increase knowledge of the collected data. Next, machine learning algorithms and statistical models are proposed (6) and applied to communicate, visualize and report findings (7). In our case, we describe a fitting algorithm that delivers input for our data centre simulation models. The outcomes of such simulations provide insight that is useful in decision-making (8) or could form a data product to assist in data centre design (9).

8.3 Data Preparation

In this section, the preparation process of the data is provided, i.e., the steps taken before any fitting algorithm is applied. This process consists of the collection of raw data that is processed and cleaned with a tool chain, which environment is also suitable for the fitting algorithm. First, the chain of tools is elaborated in Section 8.3.1. The collected raw data is described in Section 8.3.2. The processing and cleaning steps taken are provided in Section 8.3.3.

8.3.1 Tool Chain

The data preparation and fitting algorithm are implemented with a chain of tools, as follows. The code is written in PYTHON version 3.x (the original code was written using version 3.6) and is publicly accessible via a GITHUB repository [147]. The PYTHON code requires the libraries PANDAS [142] for data analysis, NUMPY [137] and SCIPY [158] for numerical algorithms, MATPLOTLIB for plotting, and some more basic libraries such as RANDOM and MATH. JUPYTER NOTEBOOK is used to document (with the lightweight markup language MARKDOWN), write (live) PYTHON code in a web-based environment and store code in the IPYNB format. A helpful and recommended data science platform to easily install JUPYTER NOTEBOOK [149], PYTHON and the required libraries is ANACONDA [6]. Other required packages are operating system dependent. The data preparation and fitting algorithm have been implemented for both WINDOWS and LINUX (tested in UBUNTU).

The code uses the external program PROFIDO [18], [167]. PROFIDO is a toolkit used to create flow graphs for the fitting and modelling of distributions for given datasets. The flow created in PROFIDO can be exported to a shell script, which then executes the PROFIDO algorithms associated with the flow graph. Shell scripts are configured in such a way that it operates on other machines without requiring any alterations in the scripts.

An overview of the application of the tool chain is provided in Figure 8.2.

Figure 8.2: Overview of the tool chain.

Here the raw data is provided as an input to the PYTHON code. The system then computes the desired plots and distributions.

8.3.2 Raw Data

Our data is collected from a commercial data centre containing 10 nodes, which main job is to respond to a special type of search queries sent to these nodes by users via the Internet. The data ranges from August 10 to August 31 (in 2016); and from September 26 to October 17 (also in 2016). The total set contains about 7.6 million (19.6 million) entries and its size is unpacked about 550 MB (1401 MB) in CSV format (about 71 MB or 211 MB as GZIP). Each entry in the data set represents a single job and is provided with information about (i) date and time of arrival, (ii) service node identifiers, (iii) HTTP request method, (iv) response size, (v) HTTP status code, and (vi) service time (in ms).

8.3.3 Processing and Cleaning Data

Additional exploratory data analysis was performed to increase our knowledge of the data set, which can also be found on our GITHUB repository [147]. The raw data is processed by transforming it into a PANDAS DataFrame, which represents a 2-dimensional labeled data structure with columns of potentially different types (cf. Figure 8.3). After the raw data is processed using the tool chain, the data needs to be *cleaned* before it can be analysed, as follows:

- Remove all error data with a duration of 0 and a response size of *NaN*.

	datetime	dataset_id	cluster_host	request_method	response_size	status_code	duration
1	2016-08-10 00:00:08.174	17	jaws09.data.be.nl	GET	135701.0	200	115
2	2016-08-10 00:00:08.450	17	jaws08.data.be.nl	GET	135701.0	200	121
4	2016-08-10 00:00:08.728	18	jaws07.data.be.nl	GET	33853.0	200	78
6	2016-08-10 00:00:09.646	1	jaws07.data.be.nl	GET	284529.0	200	90
7	2016-08-10 00:00:13.524	22	jaws09.data.be.nl	GET	77987.0	200	75

Figure 8.3: Five processed and cleaned data entries with corresponding job information.

- Remove the first entry, since the inter-arrival time of this job is unknown.
- Remove all jobs with a duration larger than a certain value; this data is removed in consultation with the company that indicated these jobs to be uncommon under regular working conditions. Solely for this reason, about 5%, respectively 9% of all entries are omitted from the oldest and newest data sets. Long service times for these uncommon jobs are caused by garbage collection, batch processes, updates, or exceptionally large bursts of jobs.

Since the data collection is tailored towards the goal of finding valuable workload characteristics, the cleaning steps taken suffice for this purpose.

8.4 High-Level Design of Algorithm

This section describes the algorithm to compute the overall distribution given a data centre workload trace. We distinguish two sequential phases. The first phase roughly estimates the distribution; whereas the second phase refines it.

8.4.1 First Phase

Algorithm 2 contains the pseudo-code for the *first phase* that computes a rough estimate of the distribution. Note that the algorithm contains pointers to Figure 8.4–8.7, to better illustrate the intermediate results.

The input data is used to construct a histogram to determine the local minima and maxima as a first step in the algorithm. These local minima and maxima allow us to determine the boundaries of one or more normal distributions. In order to describe this mixture of normal distributions, a weight w_i is computed to express the fraction of data that lies between the boundaries used to determine normal distribution i . The weight w_i for normal distribution i is computed by dividing the number of data entries $|PrepData[LB_i : UB_i]|$ between the boundaries of normal distribution i over the total number of data entries $|PrepData|$, as follows:

$$w_i = \frac{|PrepData[LB_i : UB_i]|}{|PrepData|}. \quad (8.1)$$

The random variable X has a mixture of normal distributions with probability density function $pdf_{mix}(x)$, that is computed by taking the sum of the computed

Algorithm 2: High-level pseudo-code of the *first phase* of the algorithm.

Data: PrepData : processed and cleaned data as a Pandas DataFrame
Result: The mixture of normal distributions formula

```

1 /* Figure 8.4 - Obtain histogram from data.                */
2 Hist = GETHISTOGRAM(PrepData);
3 /* Figure 8.5 - Find the local minima and maxima          */
   from histogram data.                                     */
4 Maxima = FINDLOCALMAXIMA(Hist);
5 Minima = FINDLOCALMINIMA(Hist);
6 /* Figure 8.6 - Estimate boundaries per normal            */
   distribution.                                             */
7 BoundariesList = COMPUTEBOUNDARIES(Maxima, Minima);
8 /* Figure 8.7 & Algorithm 3 - Create a mixture of         */
   normal distributions by fitting each distribution        */
   with data between the boundaries.                        */
9 return CREATEMIXTUREDISTRIBUTION(BoundariesList, PrepData);

```

Algorithm 3: High-level pseudo-code of a *helper function* that returns a mixture of normal distributions by fitting a normal distribution between chosen boundaries using a fitting tool with the prepared data.

```

1 function CREATEMIXTUREDISTRIBUTION(BoundariesList :
   list of boundaries, PrepData : prepared data) is
2   foreach Bounds  $\in$  BoundariesList do
3     LB = Bounds.GETLOWERBOUND();
4     UB = Bounds.GETUPPERBOUND();
5     Normals += FittingTool.COMPUTENORM(PrepData[LB : UB]);
6     /* Equation 8.1 - The fraction of data that            */
       lies between the boundaries.                          */
7     Weights +=  $\frac{|PrepData[LB:UB]|}{|PrepData|}$ ;
8   end
9   /* Equation 8.2 - The sum of the pdfs multiplied        */
       by a weight value.                                    */
10  MixtureDistribution = COMBINENORMALS(Normals, Weights);
11  return MixedDistribution;
12 end

```

Figure 8.4: Illustrative histogram of job service time occurrences.

Figure 8.5: Step I: The histogram with local minima and maxima.

Figure 8.6: Step II: The histogram with boundaries.

Figure 8.7: Step III: The histogram with fitting normal distributions.

probability density function $pdf_i(x)$ multiplied by the computed weight value w_i of normal distribution i for all $i \in I$, as follows:

$$pdf_{mix}(x) = \sum_{i=1}^{|I|} w_i \cdot pdf_i(x), \quad (8.2)$$

where $\sum_{i=1}^{|I|} w_i = 1$. Since all the data requires to be categorised, the algorithm is expected to find normal distributions for each part in the data. However, some parts of the distribution are nearly flat in comparison to the rest of the distribution. These parts often result in a large set of additional irrelevant normal distributions (with negligible weights). Therefore, we omitted these normal distributions, which might cause $\sum_{i=1}^{|I|} w_i \leq 1$.

The overall mean μ_{mix} and standard deviation σ_{mix} of random variable X

with a mixture of normal distributions is computed, as follows:

$$\mu_{mix} = \sum_{i=1}^{|I|} w_i \mu_i \quad (8.3)$$

$$\sigma_{mix} = \sqrt{\left(\sum_{i=1}^{|I|} w_i (\sigma_i^2 + \mu_i^2) \right) - \mu_{mix}^2} \quad (8.4)$$

The normal distributions (cf. line 5 in Algorithm 3) are computed using the PROFIDO toolkit that directly applies the Maximum Likelihood Estimator (MLE) technique [107]. An EM-algorithm could also have been applied to estimate the mixture of models with a tool like PYMIX [150], but these are not always efficient to implement (as can be seen in [55]). In order to speed up computation of the normal distribution, samples are extracted deterministically from the input data with 1 sample after every 1000 data entries. Although the distribution alterations result in a slightly different fit, it significantly improves the computation time spent by PROFIDO. This form of *systematic sampling* has the main advantages of (i) simplicity compared to other statistical sampling methods, and (ii) even sampling of the data, that allows detect groups. On the other hand, the main disadvantage is the potential misrepresentation of data with periodicity. Yet, other well-known statistical sampling methods, e.g. simple random, stratified and cluster sampling, could improve the overall representation of the data. We consider the well-known systematic sampling method to represent the data well enough for its intended purpose.

8.4.2 Second Phase

The *second phase* of the algorithm refines the distribution. In this phase, the normal distributions are compared, using a goodness of fit metric, to alternative normal distributions with changed boundaries. Algorithm 4 contains pseudocode for the second phase. The boundaries are adapted as long as the goodness metric improve (lines 1–7). Subsequently, the means and standard deviations are adapted as long as the goodness metric improve (lines 8–14). The goodness of fit metric allows to study the difference between observed and expected frequencies, hence, decide how well the mixture of normal distributions fit the actual data set.

The goodness of fit metric is based on the well-known *Pearson chi-squared test*, that uses the chi-square value $\chi^2 = \sum_{i=1}^N \frac{(O_i - E_i)^2}{E_i}$ as an indication of the

Algorithm 4: High level pseudo-code of the *second phase* of the algorithm

Data: PrepData : processed and cleaned data as a Pandas DataFrame,
 MD : The mixture of normal distributions from the first phase (cf. Algorithm 2)

Result: The *refined* mixture of normal distributions formula

```

1 /* Refine each normal distribution by changing the
   boundaries */
2 AlteredMD = MD;
3 while AlteredMD.HASBETTERFIT(MD) do
4   MD = AlteredMD;
5   /* Algorithm 3 - Computes the mixture of normal
      distributions with adjusted boundaries */
6   AlteredMD = CREATEMIXTUREDISTRIBUTION(MD.ADJUST-
      BOUNDARIESLIST(),PrepData);
7 end
8 /* Refine each normal distribution by changing the
   mean and standard deviation */
9 AlteredMD = MD;
10 while AlteredMD.HASBETTERFIT(MD) do
11   MD = AlteredMD;
12   /* Equation 8.2 - The sum of the adjusted
      probability density functions multiplied by a
      weight value. */
13   AlteredMD =
      COMBINENORMALS(MD.ADJUSTNORMALS(),MD.GETWEIGHTS());
14 end
15 return MD;
```

goodness of fit, where O_i is the observed frequency, E_i the expected frequency and N is the number of service times bins. The chi-squared test suffices in many situations. However, the number of samples in our data sets is too large to obtain a statistically significant fit, i.e., when the distribution is off by a relatively small margin, the goodness of fit will change significantly. In order to still give an useful indication with our large number of samples over the entire range of service times, the alternative goodness of fit value $\sum_{i=1}^N (O_i - E_i)^2$ is computed using the same parameters for frequencies and bins (O_i , E_i and N) as with the chi-squared test. This alternative goodness of fit value suffices to observe improvements, while the effect of small sized samples is cancelled.

With some experiments, we found that our goodness of fit value results in a mean and variance slightly closer to the observed values and significantly better estimates of the peaks. Our goodness of fit value is different in that it considers an accurate estimation of the mean and variance to be more important than accurately fitting the lower frequencies to the actual data.

The algorithm implementation and a more detailed elaboration thereof is provided in the GITHUB repository [147].

8.5 Experimental Validation

The algorithm is validated with synthetic traces to discuss specific algorithmic details in Section 8.5.1, before it is used with real-world traces in Section 8.5.2. In Section 8.5.3, we show the use of the fitted data in a small data centre simulation study.

Throughout this section, the tables are provided with weights, means and standard deviations to form a mixture of normal distribution based on (8.2). Furthermore, we provide the mean and standard deviation of the mixture distribution based on (8.3).

8.5.1 Generated Traces

Traces are generated by creating data entries with samples taken from a mixture of normal distributions. Subsequently, the algorithm is then used to find the mixture of normal distributions based on the generated trace. From a set of randomly generated mixture of normal distributions, three distributions are hand-picked in order to pinpoint the main challenges of the algorithm, and elaborated below. Note that in most cases the algorithm finds very accurate fits for the distributions. The computation times of the algorithm for the three

case	phase 1	phase 2a	phase 2b	total
I	4 s	12 s (=4 s/iteration)	38 s (=3.8 s/iteration)	54 s
II	2 s	16 s (=5 $\frac{1}{3}$ s/iteration)	39 s (=3.9 s/iteration)	57 s
III	2 s	18 s (=6 s/iteration)	44 s (=4.4 s/iteration)	64 s

Table 8.1: Total computation times of the algorithm for the three artificially generated traces and its subdivision of the computation times into the phases 1 (Algorithm 2), 2a (lines 1–7 from Algorithm 4) and 2b (lines 8–14 from Algorithm 4).

Figure 8.8: Service Times Histogram and Fit of Generated Distribution I with Undetected Overlapping Peaks.

weight w_i	mean μ_i	st.dev. σ_i
0.345	148.997	4.058
0.204	189.550	4.770
0.267	272.070	5.852
0.065	287.848	4.490
0.067	297.278	4.029
0.051	309.463	5.317
$\sum_{i=1}^{ I } w_i$	μ_{mix}	σ_{mix}
= 1.00	= 217	= 62.8

Table 8.2: Normal Distributions Mixture Values of the *Actual* Generated Distribution I.

weight w_i	mean μ_i	st.dev. σ_i
0.350	148.363	4.071
0.207	189.000	4.817
0.269	271.721	6.102
0.135	295.275	8.616
0.040	307.467	12.946
$\sum_{i=1}^{ I } w_i$	μ_{mix}	σ_{mix}
= 1.00	= 216	= 61.3

Table 8.3: Normal Distributions Mixture Values of the *Fitted* Generated Distribution I.

artificially generated traces are provided in Table 8.1. Note that, the computation time of the second phase is considerably slower than the first phase. This is mainly because the second phase has to recompute the mixture of normal distributions multiple times with the fitting tool (Algorithm 3 and line 6 from Algorithm 4), while the first phase only compute the mixture of normal distributions once (line 9 from Algorithm 2).

Case I: Undetected Overlapping Peaks.

The first distribution is shown as a histogram in Figure 8.8. In general, these figures display blue bars of a histogram, that depict the number of job occurrences on the y -axis and correspond to a specific service time range as indicated on the x -axis, whereas the green line corresponds to the estimated mixed normal distribution (cf. Figure 8.7). The corresponding values of the mixed

Figure 8.9: Service Times Histogram and Fit of Generated Distribution II with Hidden Distributions.

weight w_i	mean μ_i	st.dev. σ_i
0.309	190.649	20.755
0.236	162.087	15.709
0.310	241.905	22.306
0.145	268.929	21.216
$\sum_{i=1}^{ I } w_i$ = 1.00	μ_{mix} = 211	σ_{mix} = 43.4

Table 8.4: Normal Distributions Mixture Values of the *Actual* Generated Distribution II.

weight w_i	mean μ_i	st.dev. σ_i
0.597	175.565	26.053
0.403	246.983	23.340
$\sum_{i=1}^{ I } w_i$ = 1.00	μ_{mix} = 204	σ_{mix} = 43.0

Table 8.5: Normal Distributions Mixture Values of the *Fitted* Generated Distribution II.

normal distributions can be found in Table 8.2–8.3. This example exposes one of the weaknesses of the algorithm. In the plot and the tables, only five of the six normal distributions are detected. The two smaller distributions have such small peaks that these remain undetected in the algorithm. The consequence is that the fitting merges the two normal distributions into one normal distribution. When observing the other peaks in the plot, the algorithm shows very accurate fits of the normal distributions. The small differences between the means and standard deviations corresponding to the same normal distribution can be explained by the generation of traces with samples taken from the normal distribution.

Figure 8.10: Service Times Histogram and Fit of Generated Distribution III with Overestimated Overlapping Peaks.

weight w_i	mean μ_i	st.dev. σ_i
0.333	40.000	8.000
0.333	60.000	8.000
0.333	80.000	8.000
$\sum_{i=1}^{ I } w_i$	μ_{mix}	σ_{mix}
= 1.00	= 60.0	= 18.2

Table 8.6: Normal Distributions Mixture Values of the *Actual* Generated Distribution III.

weight w_i	mean μ_i	st.dev. σ_i
0.326	40.946	7.952
0.355	59.941	8.471
0.319	78.622	7.759
$\sum_{i=1}^{ I } w_i$	μ_{mix}	σ_{mix}
= 1.00	= 59.7	= 17.2

Table 8.7: Normal Distributions Mixture Values of the *Fitted* Generated Distribution III.

Case II: Hidden Distributions.

The second distribution is shown as a histogram in Figure 8.9. The corresponding values of the mixed normal distributions can be found in Table 8.4–8.5. This distribution illustrates the issue of “hidden distributions”, that remain undetected in the algorithm. This distribution is a mixture of four normal distributions, however, the algorithm only detects a mixture of two. As can be seen in the plot and tables, the means and standard deviations are so close to each other that the pairs of underlying normal distributions are difficult to distinguish without prior knowledge. Since the algorithm detects only one maximum and two minima in such a case, two underlying normal distributions remain undetected.

Case III: Overestimated Overlapping Peaks.

The third distribution is shown as a histogram in Figure 8.10. The corresponding values of the normal distributions can be found in Table 8.6–8.7. This mixture of distributions is a simple example of three normal distributions with identical shapes that clearly overlap, i.e., the mixture distribution consists of three normal distributions with shifted means, equal standard deviations and equal weights with overlapping peaks. Although the algorithm captures the form of the distribution well, it overestimates the frequencies at the extrema between the normal distributions. A plausible explanation is that the distributions are adjusted separately during the refinement process.

8.5.2 Real Data Centre Traces

The algorithm is also provided with two processed and cleaned data sets (cf. Section 8.3). These data sets have been recorded at different times and are intuitively referred to as $\mathcal{A}$ for the oldest dataset and $\mathcal{B}$ for the newest dataset. Both of these data sets are obtained over a three-week period. $\mathcal{A}$ contains about 5 million entries after data preparation, whereas $\mathcal{B}$ contains about 15 million useful entries. The service time of each entry is used to construct histograms. Total computation times of the algorithm for $\mathcal{A}$ and $\mathcal{B}$ are, respectively, 137 s (7 s without refinement) and 545 s (29 s without refinement).

Figure 8.11 shows the service times histogram and resulting fit after execution of the algorithm on the $\mathcal{A}$. The corresponding values of the normal distributions can be found in Table 8.8. As can be seen by comparison of the curve of the fitting (green line) and histogram of the data (blue bars), the curve clearly follows the distribution pattern.

Figure 8.11: Service Times Histogram and Fit of the $\mathcal{A}$ Distribution.

Figure 8.12: Service Times Histogram and Fit of the $\mathcal{B}$ Distribution.

			weight w_i	mean μ_i	st.dev. σ_i
weight w_i	mean μ_i	st.dev. σ_i	0.090	3.648	0.795
0.060	3.725	0.852	0.142	29.249	8.965
0.096	21.889	2.181	0.089	74.991	9.484
0.232	30.111	2.554	0.120	84.201	9.790
0.313	81.995	8.129	0.239	107.169	13.481
0.174	117.495	11.066	0.063	142.874	9.241
0.048	164.879	15.191	0.031	160.354	7.756
0.051	194.997	19.979	0.070	177.298	9.203
0.027	211.150	14.388	0.062	197.853	10.826
			0.095	227.669	15.100
$\sum_{i=1}^{ I } w_i$	μ_{mix}	σ_{mix}	$\sum_{i=1}^{ I } w_i$	μ_{mix}	σ_{mix}
= 1.00	= 79.0	= 55.3	= 1.00	= 107	= 67.0

Table 8.8: Normal Distribution Mixture Values of the $\mathcal{A}$ Distribution.

Table 8.9: Normal Distribution Mixture Values of the $\mathcal{B}$ Distribution.

Figure 8.12 shows the service times histogram and resulting fit after execution of the algorithm on the $\mathcal{B}$. The corresponding values of the normal distribution can be found in Table 8.9. Also with this data set, the curve of the fitted data clearly follows the distribution pattern. Similar to the cases discussed in Section 8.5.1, the plot shows some small overlapping peaks that remain undetected (e.g., around 10 ms, 30 ms and 50 ms) and frequencies at local extrema are overestimated (e.g., around 90 ms). On the right side of the plot, the peaks of the histogram are at different places than the peaks of the fit. This is mainly caused by the detection of multiple normal distributions in a relatively small area that causes a lot of overlap between them.

Although in both cases the fit is not perfect, it does follow the trend of the data. Therefore, we conclude that the choice of normal distributions was good. We expect that this type of distribution generally works well when fitting service times of a system where each request corresponds to a database query. This stems from the fact that there are generally many different types of queries, each with their own mean service time.

8.5.3 Comparison with Calibrated Models

In this section, we compare exponentially distributed service times with the mixture of normal distributions for which the values are taken from the $\mathcal{B}$ distribution from Table 8.9.

For this purpose, the data centre simulation framework DACSIM (as proposed in Chapter 4) is executed with these service times, as detailed below. The simulation framework is configured such that it simulates a data centre with a cluster of 150 dual-processor blade servers inside DELL POWEREDGE server racks, i.e., a total of 300 CPU cores. There is a base power consumption for each rack of 172.8 W and a power consumption of 145.5 W for each active CPU. For both simulation configurations, jobs arrive according to a Poisson process such that the inter-arrival times distribution is exponential with λ set to approximately 1121 jobs/s (20% active CPUs), 1682 jobs/s (30% active CPUs), and 2242 jobs/s (40% active CPUs). The mean service time of the exponentially distributed variant is set equal to that of the $\mathcal{B}$ distribution, which is 107 ms. Jobs are scheduled in round-robin fashion to the racks, which are then again scheduled in round-robin fashion to the available servers in that rack. The simulation uses an existing power management strategy that uses exponentially moving averages of the response times and the time spent in particular power states to determine which servers should be sleeping and off. More details on the used data centre configuration, power management strategy and simulation models

Figure 8.13: Mean power consumption of a simulated data centre of (i) exponentially distributed and (ii) $\mathcal{B}$ distributed service times for three processing utilisations.

can be found in Chapter 4–6.

Figure 8.13 shows that the mean power consumption for the data centre with the $\mathcal{B}$ distributed service times is larger. The figure shows a difference between the mean power consumptions of 0.685 kW (1.42 %), 1.191 kW (1.86 %) and 2.272 kW (2.88 %) for, respectively, the processing utilisations 20%, 30% and 40%. As the exponentially distributed service times is often assumed, these results show the relevance of using appropriate job characteristics for the estimation of power consumption in data centres.

8.6 Conclusions

In this chapter, we have proposed a data science approach towards the characterisation of workloads in data centres. We have successfully applied our algorithm on a data trace of about 20 million entries of a real data centre to fit to a mixture of normal distributions. Our algorithm first roughly estimates a mixture of normal distributions by detecting local extrema in the histogram of the data and building normal distributions around these. It subsequently improves these distributions by altering the data bounds, mean and standard deviations as long as suitable chosen goodness of fit measure improves. Additionally, using three artificially generated traces, the main challenges for the algorithm are found: undetected overlapping peaks, hidden distributions, and overestimating overlapping peaks. Finally, an example data centre is simulated with models calibrated with these service time distributions, which shows relevant differences in the mean power consumption.

Cross-Model Validation

In this chapter, two implementations for power and performance analysis in data centres with load balancing techniques and power management (PM) functionality are compared for validation purposes. The results from the simulation framework DACSIM, as introduced in Chapter 4, are compared with the results from an implementation in IDSL, a language and tool-chain for evaluating service-oriented systems. As both these implementations use different modelling and analysis techniques, both require, for comparison purposes, (i) a clear system description for the development and configuration of the power and performance models, and (ii) an effective way to describe and process job dispatching policies specifically for the practice of load balancing. This chapter introduces a syntax for the description of job dispatching policies. For both implementations, this syntax allows us to discuss multiple job dispatching policies. Adjustment of time-outs and queue size thresholds in the policies lead to a power-performance trade-off and successful validation by comparison between the two implementation.

This chapter is organised, as follows. First, an introduction is provided in Section 9.1. Section 9.2 provides the system description. Section 9.3 formalises load balancer in terms of power and performance together with a syntax for job dispatching policies. Section 9.4 then provides two implementations of power and performance models. Our experimental results are compared in Section 9.5. Section 9.6 concludes the chapter.

This chapter is based on the publication [B5], which also served as basis for a chapter in the PhD thesis of Freek van den Berg [20].

9.1 Introduction

Recall that *advanced PM* allows to switch between power states of servers to reduce power consumption, while trying to keep the performance intact (e.g., bringing to sleep underutilised servers). There are two key elements required to construct an effective PM strategy, namely: (i) *power state switching* and (ii) *job dispatching*. PM strategies describe when servers should switch between the power states. Subsequently, the load should be balanced among the servers such that optimal performance is obtained via the dispatcher.

This chapter proposes a powerful, yet concise, policy language, which covers, among others, policies that observe the size of the queue to decide to which server jobs are assigned. Furthermore, servers are put to sleep when idling with a simple time-out mechanism, which should be easy to implement in actual servers as literature suggests [19]. The method proposed in this chapter allows us to explore a large set of designs by adjusting only three parameters: (i) *queue size threshold*, (ii) *idle time-out*, and (iii) *non-determinism resolution*. In the end, this leads to interesting power-performance trade-offs, i.e., the possibility to exchange power consumption reductions at the cost of performance.

The above policies are implemented as extensions to IDSL (as introduced in [22]) and DACSIM (as introduced in Chapter 4) so that policies can be automatically evaluated. This provides insights in the effectiveness of the policies with respect to power and performance. The implementations describe the same system with different types of models. Therefore, we perform a *cross-model validation*, i.e., different modelling and analysis approaches are validated.

In order to do so, we first extend our model to support job dispatching policies for load balancing. We extend the DACSIM simulation framework from Chapter 4 to support a powerful, yet concise, policy language, that covers policies that observe, among others, the size of the queue to decide to which server jobs are assigned.

With the ability to describe job dispatching policies, we then configure the models of both implementations with a (simplified) data centre performance cluster for which we will study the potential of load balancing techniques and PM strategies. We construct a small example consisting of four servers, which is a simplified representation of a realistic performance cluster that allows for job dispatching policies to direct load based on state (information) variables. This example suffices for discovering relevant insights, while computation times are still reasonable for both implementations (especially for the, to this case yet less optimised, IDSL implementation).

For this case study, we then evaluate many job dispatching policies to find

the policies that are good with regard to both energy and performance. We evaluate a large set of policies that are a composition of three variables, viz., (i) *queue size threshold*, (ii) *idle time-out*, and (iii) the way of resolving (remaining) *non-determinism*. With these three degrees of freedom, we explore the trade-offs between the average response time and the average power consumed by all servers.

Moreover, we provide a comparison of the results obtained from the two implementations by comparison of performance. We accomplish model validation by analysis of performance for many designs using DACSIM, and comparing the outcomes of a supposedly equivalent implementation in IDSL (as elaborated in [20], [B5]). The generic language and toolchain for evaluating service-oriented systems IDSL has been extended to support analysis in a similar context.

There are many more modelling and analysis approaches that discuss power and performance aspects (as shown earlier in Table 2.2). The work of [144], [182] also discuss power-performance trade-offs that exists with regard to virtual machines. In [144], this has been elaborated with a testbed to apply their models for monitoring and control. Furthermore, [145] takes cluster-based systems into account with a load balancer for power and performance analysis. Of course, there are also many other tools available for analysis of data centres with power and performance aspects that are already mentioned in Section 1.2. Alternatives for IDSL, that offer similar high-level languages, are PRISM [80], MODEST [74] and UPPAAL [104]. CLOUDSIM [31] is an alternative for DACSIM.

Our work distinguish itself in the following three ways: First, we have constructed a load balancer policy with a powerful yet concise language which is used to access system variables, such as queue sizes and power states of servers. Second, we have implemented this policy in two different development environments, IDSL and DACSIM. Both implementations were validated by comparison of evaluated results. Third, evaluation of many policy designs provides insight in the meaning of the policies while only using two parameters: (i) The queue size threshold that affects the power-performance trade-off, and (ii) the server idle time that affects the level of Pareto optimality.

9.2 System Description

In this section, a system description is provided for a data centre with emphasis on load balancing. Recall from Section 2.4 that load balancing involves the im-

proved distribution of tasks among various computing resources, e.g., servers, for better power and performance. A smart job dispatcher with load balancing techniques in combination with PM features offer great opportunities to reduce energy consumption, while performance is kept intact. In Section 9.2.1, we describe a simplified case study based on a data centre performance cluster. Section 9.2.2 describes the metrics that provide an insight in the effectiveness of a job dispatching policy.

9.2.1 Case Study: A Data Centre Performance Cluster

The case study in this chapter has been inspired by the so-called PEREGRINE cluster at the CENTER FOR INFORMATION TECHNOLOGY (CIT) [176] in Groningen, the Netherlands, as follows. Assume the CIT decides to actively use PM features for their PEREGRINE computational cluster [177] in combination with the appropriate load balancing techniques to improve the overall performance. This cluster has a total of 4368 cores with three types of nodes, namely: (i) 162 standard nodes with 2×24 INTEL XEON 2.5 GHz cores; (ii) 6 standard nodes equipped with accelerator cards; and (iii) 7 fast nodes with 4×48 INTEL XEON 2.6 GHz cores. Each standard node consumes approximately 40 W for only the CPU cores [88].

This system has many details with regard to power and performance that are either unnecessary to be modelled, or still publicly unknown/inaccessible. Therefore, the system is simplified and supplemented with realistic assumptions to the best of our knowledge. Moreover, simplifications are necessary to reduce the overall complexity especially for computation speed in the yet less optimised IDSL implementation. For both implementations to represent similar systems, we state our joint assumptions explicitly. For these reasons, we model the system using the following list of assumptions:

- Incoming jobs arrive according to a Poisson process with a negative exponential distribution with rate 1 job per second;
- The system consist of four similar resources (or server nodes);
- A job dispatching policy specifies how incoming jobs are distributed over these four servers;
- Servers have four power states each; they are either switched on ("stateOn"), asleep ("stateAsleep"), moving from "stateOn" to "stateAsleep" ("stateSleeping"), or moving from "stateAsleep" to "stateOn" ("stateWakeup");
- A server can only process tasks in state "stateOn";

- Servers spend exactly 10 seconds in transition states “stateSleeping” and “stateWakeup” each, when changing states;
- Servers each consume 200 Watt in states “stateOn”, “stateSleeping” and “stateWakeup”, and 14 Watt in state “stateAsleep”, based on empirical studies [13], [66];
- Servers have infinite queues that adhere to a non-preemptive FCFS scheduling policy;
- Servers process incoming jobs deterministically with rate 1 job per second;
- Evaluating a job dispatching policy takes *no* time, i.e., *no* computational resources are required for evaluation of job dispatching policies.

9.2.2 Policy Effectiveness

In this chapter, the effectiveness of a job dispatching policy for load balancing is determined in terms of electrical power consumed and performance.

Power The way a policy distributes incoming service jobs indirectly affects power consumption, i.e., when a policy does *not* distribute jobs to any server for a specified amount of time, the server will go to sleep and use only a fraction of power. In this chapter, we consider the average power (in Watt) that the four servers use *together*.

Performance A policy distributes incoming jobs over a number of servers. The way of distributing strongly affects multiple performance metrics, e.g., the queue sizes and utilisation of a specific resource are generally high when the load balancer distributes many jobs to the same server. In turn, this increases the response times for jobs that are processed by this server. In this chapter, only the (average) response time is considered, because it is a well-known metric to indicate performance.

9.3 Power and Performance Model Assumptions

In order to be able to reason about the same system, we describe high-level power and performance models for data centres with load balancing techniques to serve as a basis for both model implementations. Thereby allowing us to make a good comparison between different implementations at a later time.

In the previous section, we described the system assumptions of a data centre with a job dispatcher for load balancing. Here, we provide a high-level

model to evaluate power and performance of job dispatching policies for load balancing. Section 9.3.1 defines power and performance by considering incoming jobs, power states and transitions, and response times of jobs. Section 9.3.2 specifies job dispatching policies using a grammar and semantics, some typical examples, as well as a number of mechanisms to resolve non-determinism.

9.3.1 Characteristics for Power and Performance

In this section, we define the power and performance characteristics of job dispatchers for load balancing in three aspects. We consider the load as a stream of jobs according to the basic model as depicted in Figure 1.3 by describing the jobs and their distribution among the computational resources. Subsequently, we describe how PM interacts based on the jobs in the system. Furthermore, we clarify how the average power and performance metrics are computed.

Job Arrival Process and Distribution

Jobs arrive with an exponentially distributed inter-arrival time, with rate 1 job per second at the job dispatcher. The incoming jobs are inspected by the job dispatcher, which distributes the jobs over selected servers for processing, based on a pre-defined policy.

Example 9.1 (Illustrative distribution of arriving jobs). Let $I(t)$ indicate that a job arrived at time t . jobs have a unique arrival time. Then $I : 2^{\mathbb{R}}$ is an infinite set with the arrival times of all incoming jobs. For illustration, the following I has been generated using a random number generator:

$$I = \{0.87, 0.91, 1.46, 2.03, 3.54, 4.68, 5.42, 5.52, 5.66, 7.26, 9.61, 10.34, \dots\}. \quad (9.1)$$

Let $\mathcal{P}(t) : \mathbb{R}^+ \rightarrow \{S_1, S_2, S_3, S_4\}$ be a load balancer policy that distributes the job that arrived at time t to either server S_1, S_2, S_3 or S_4 . Let $S_m = \{t \in I | \mathcal{P}(t) = S_m\}$ be the incoming jobs of server m . Hence, $\{S_1, S_2, S_3, S_4\}$ is a partition of I , for illustration, assume policy $\mathcal{P}$ distributes the above incoming job I over the four servers, as follows.

$$\begin{aligned} S_1 &= \{0.91, 3.54, 5.42, \dots\}, & S_2 &= \{5.66, 9.61, 10.34, \dots\}, \\ S_3 &= \{2.03, 4.68, 7.26, \dots\}, & S_4 &= \{0.87, 1.46, 5.52, \dots\}. \end{aligned} \quad (9.2)$$

□

Figure 9.1: Power states of resources and their transitions

Power Model Assumptions

Recall from Section 2.3.2 that PM allows servers to switch from one power state to another. While switching power states comes with a certain delay and power consumption varies between power states, electrical energy can be saved if done properly. In this chapter, servers are in exactly one out of four power states at a time, viz., on (*on*), sleeping (*sl*), asleep (*as*), or wakeup (*wu*).

Figure 9.1 visualises the four power states (central nodes) used in this chapter. Also, the figure illustrates the five transitions, as follows:

- (1) When a server becomes and remains idle for the next TO seconds, the server stays on for TO seconds (Figure 9.1(1a)), is busy with putting the server to sleep for the next 10 seconds (1b), and ends in sleep mode (1c).
- (2) When a new job arrives at a server in sleep mode (Figure 9.1(2a)), it is waking up for the next 10 seconds (2b), and then is turned back on and start processing the job (2c).
- (3) When a new job arrives at a server while it is putting a server to sleep (Figure 9.1(3a)), it finishes putting the server to sleep and then directly

starts waking up again for the next 10 seconds (3b), and is turned on (3c) eventually.

- (4) When a server is in sleep mode (Figure 9.1(4)), it remains there as long as *no* new jobs arrive.
- (5) When a server is turned on (Figure 9.1(5)), it remains turned on as long as new jobs arrive before TO seconds are expired in an idling server.

Power and Performance Metrics

Section 9.2.2 addressed performance and energy consumption as the properties to evaluate a load balancer on. We reduce the complexity of the system by considering only the mean power consumption for the length of the simulation, as formally defined in Equation (4.1), and the mean response time, as formally defined in Equation (4.5). For the computation of the mean power consumption, we consider the power consumption rewards as depicted in Figure 9.1 for a number of four servers without considering power consumed by any infrastructural components, such that formally $P_{\text{other}}(t) = 0$ in the formula.

9.3.2 Specification of Job Dispatching Policies

A load balancer policy prescribes how a load balancer behaves with respect to distributing incoming jobs to servers. First, we define a policy using a mechanism that orders the servers; Next, we present some example policies. Finally, we provide ways to resolve non-determinism when a policy is ambiguous.

Policy Syntax

Each time an incoming job arrives, a load balancer has to select one of the servers to delegate this job to. We use the following algorithm to make this decision:

- Relevant system state variables are retrieved, e.g., the queue sizes of the servers.
- The preference of each server is determined via an arithmetic expression that includes system state variables.
- The incoming job is delegated to the most desirable server, viz., the server with the highest outcome for the arithmetic expression.

Figure 9.2 shows the grammar of a policy expression $\mathcal{P}$. Policy expression $\mathcal{P}$ can express in the categories *state*, *power*, *time* or *math*, as follows.

$\langle \mathcal{P} \rangle$	::=	$\langle \text{ID} \rangle \mid \langle \text{numServers} \rangle \mid \langle \text{queueSize} \rangle \mid \langle \text{state} \rangle$ $\mid \langle \text{power} \rangle \mid \langle \text{time} \rangle \mid \langle \text{math} \rangle$
$\langle \text{state} \rangle$	::=	$\langle \text{stateOn} \rangle \mid \langle \text{stateAsleep} \rangle \mid \langle \text{stateSleeping} \rangle$ $\mid \langle \text{stateWakeup} \rangle$
$\langle \text{power} \rangle$	::=	$\langle \text{powerOn} \rangle \mid \langle \text{powerAsleep} \rangle \mid \langle \text{powerSleeping} \rangle$ $\mid \langle \text{powerWakeup} \rangle$
$\langle \text{time} \rangle$	::=	$\langle \text{timeWakeup} \rangle \mid \langle \text{timeSleeping} \rangle \mid \langle \text{timeOutTime} \rangle$
$\langle \text{math} \rangle$	::=	$\langle \mathcal{P} \rangle * \langle \mathcal{P} \rangle \mid \langle \mathcal{P} \rangle + \langle \mathcal{P} \rangle \mid \langle \mathcal{P} \rangle - \langle \mathcal{P} \rangle \mid \langle \mathcal{P} \rangle / \langle \mathcal{P} \rangle \mid \langle \mathcal{P} \rangle \bmod \langle \mathcal{P} \rangle$ $\mid \langle \text{INT} \rangle \mid \langle \text{RANDOM} \rangle \mid \langle \text{DSPACE} \rangle$

Figure 9.2: The grammar of load balancer policy expression $\mathcal{P}$.

- `state` provides indicator functions to check whether a server is in one of the four states, or not, e.g., when a server is in the On state, `stateOn` yields 1 and the others 0.
- `power` is used to retrieve the power consumptions of each individual state (see assumptions in Section 9.2.1).
- `time` includes `timeWakeup` and `timeSleeping`, the time it takes for the server to go back and forth between states on and sleep (see assumptions in Section 9.2.1), as well as `timeOutTime`, the time of inactivity the server undergoes before going to sleep.
- `math` provides five recursive functions that combine policies via arithmetic operations to create arbitrarily complex polices. Furthermore, a constant integer number `INT`, a random number `RANDOM` $r \in [0 : 1]$, and a design dependent variable `DSPACE` could be used.
- `ID` is a unique number for identifying servers.
- `numServers` is the total number of servers.
- `queueSize` is the number of jobs in the queue of the server.

Let the generic policy $\mathcal{P}_q$ be defined as follows:

$$\mathcal{P}_q = -\text{queueSize} - q \cdot (1 - \text{stateOn}), \quad (9.3)$$

where q is the server queue size at which an additional server is switched on; q is design dependent; and $q \geq 0$.

Example 9.2 (Generic job dispatching policy examples). To illustrate the functioning of the generic job dispatching policy $\mathcal{P}_q$ in practice, policy $\mathcal{P}_0$ assigns incoming jobs to the server that currently has the shortest queue size:

$$\mathcal{P}_0 = -\text{queueSize} \quad (9.4)$$

Table 9.1 shows an example evaluation of $\mathcal{P}_0$, where lb_{select} is the choice of the load balancer for a certain server, # n incoming job n , and the numbers in the table the policy evaluations of servers 1-4 for incoming job # n . For the sake of simplicity, we assume that *no* incoming job finished processing (yet). For jobs #1-#4, the load balancer arbitrary selects servers, because multiple servers have the highest value for the policy evaluation. In later steps this pattern is repeated. Hence, jobs are equally distributed over the servers.

$\mathcal{P}_5$ is also policy that primarily assigns new incoming jobs to the server with the shortest queue size, as follows:

$$\mathcal{P}_5 = \text{queueSize} - 5 \cdot (1 - \text{stateOn}) \quad (9.5)$$

However, $\mathcal{P}_5$ also considers the power state of the servers to save energy. Concretely, it will only switch on a new server if all currently switched-on servers have a queue size of at least 5. Note that this policy might perform less well than $\mathcal{P}_0$, however, at the benefit of reduced energy consumption.

Table 9.2 show an example evaluation of $\mathcal{P}_5$. When job #1 arrives, the policy evaluation of all servers yields -5 because *no* servers are turned on. When the load balancer arbitrarily delegates this job to server 1, server 1 switches on and its policy evaluates to -1. Consequently, the load balancer selects server 1 for jobs #2-5. Then jobs #6-7 are delegated to server 2 for similar reasons. Note that after 7 incoming jobs only two servers have received jobs, which is a good property when energy consumption is of concern. $\square$

	#1	#2	#3	#4	#5	#6	#7
0	0	0	0	0	-1	-1	-1
1	0	0	-1	-1	-1	-2	-2
2	0	0	0	-1	-1	-1	-2
3	0	-1	-1	-1	-1	-1	-1
lb_{select}	3	1	2	0	1	2	3

Table 9.1: Example evaluation of performance-optimising policy $\mathcal{P}_0$

	#1	#2	#3	#4	#5	#6	#7
0	-5	-5	-5	-5	-5	-5	-5
1	-5	-1	-2	-3	-4	-5	-5
2	-5	-5	-5	-5	-5	-5	-1
3	-5	-5	-5	-5	-5	-5	-5
lb_{select}	1	1	1	1	1	2	2

Table 9.2: Example evaluation of reasonably energy-efficient policy $\mathcal{P}_5$

	#1	#2	#3	#4
0: $\mathcal{P}'$	0.61	0.78	0.05	0.68
1: $\mathcal{P}'$	0.46	0.09	0.22	0.79
2: $\mathcal{P}'$	0.70	0.12	0.93	0.15
3: $\mathcal{P}'$	0.76	0.39	0.51	0.66
lb_{select}	3	0	2	2

Table 9.3: Example evaluation of non-determinism resolution mechanism random policy
 $\mathcal{P}' = \text{RANDOM}$

	#1	#2	#3	#4
0: $\mathcal{P}''$	0	0	0	0
1: $\mathcal{P}''$	0.25	0.25	0.25	0.25
2: $\mathcal{P}''$	0.5	0.5	0.5	0.5
3: $\mathcal{P}''$	0.75	0.75	0.75	0.75
lb_{select}	3	3	3	3

Table 9.4: Example evaluation of non-determinism resolution mechanism fixed order policy
 $\mathcal{P}'' = \text{ID}/\text{numServers}$

Resolving Non-Determinism

In this section, we provide a solution that prevents an arbitrary selection of a server by the load balancer when multiple servers have the same highest value for their policy expression.

Table 9.2 shows an example evaluation for $\mathcal{P}_5$. For each of the incoming jobs #1-#8, the load balancer selects the server with the highest evaluated value, e.g., for job #2 server 1 gets selected because its policy expression evaluates to -1, which is higher than the -5 of the other three servers. However, there are cases in which the expression of multiple servers has the highest evaluation, e.g., for job #1 all servers evaluate to 1, which makes selecting server 1 an arbitrary decision. In these cases, the load balancer performs a so-called non-deterministic decision.

Non-determinism can be resolved by adding fractions $f \in [0 : 1)$ to policy outcomes, as follows. Let $\mathcal{P}$ be a policy that only returns whole numbers $\mathbb{N}$, then policies

$$\mathcal{P}'_q = \mathcal{P}_q + \text{RANDOM} \qquad \mathcal{P}''_q = \mathcal{P}_q + \frac{\text{ID}}{\text{numServers}} \quad (9.6)$$

yield unique real numbers $\mathbb{R}$ for each server, eliminating non-determinism. The policy outcomes are only unique, if we assume that randomly drawn numbers are unique and if each server has a unique ID.

Example 9.3 (Policies resolving non-determinism). We illustrate how these two resolution mechanisms work by providing two example evaluations for them,

respectively. For the sake of simplicity, we use policies $\mathcal{P}' = \text{RANDOM}$ and $\mathcal{P}'' = \frac{\text{ID}}{\text{numServers}}$.

Table 9.3 shows how $\mathcal{P}'$ functions. For each incoming job, four random numbers are drawn and the load balancer delegates the job to the server with the highest value, e.g., job #1 is delegated to server 3, because $\max(0.61, 0.46, 0.70, 0.76) = 0.76$.

Table 9.4 shows how $\mathcal{P}''$ functions. For each incoming job, the policy evaluates to a unique number per server, which is divided by 4, the number of servers, to return number in range $[0, 1)$. The load balancer delegates all jobs to server 3 that has the highest ID, namely 3. $\square$

Design Space for Comparison

We define a design space to compare multiple policies in both implementations. Each design represents a unique combination of a job dispatching policy and a PM strategy. The design space is defined as the Cartesian product over the following three dimensions and their ranges.

- queue size threshold: $q \in \{1, 2, 3, 5, 7, 10, 15, 20, 30, 40\}$ (cf. equation 9.3).
- time-out: $TO \in \{1, 2, 3, 4, 5, 7.5, 10, 15, 30\}$.
- non-determinism resolution: $nd \in \{\text{random}, \text{fixed_order}\}$ (cf. equation 9.6).

In this design space, the job dispatcher policy chosen is either $\mathcal{P}'_q$ or $\mathcal{P}''_q$ depending on the non-determinism resolution and the value of q is set to the desired queue size threshold.

Whereas the job dispatcher is concerned with the distribution of jobs, PM is concerned with switching of power states. Therefore, a PM strategy is defined to operate next to the job dispatching policy. A simple PM strategy Θ_{val} is formulated, using the specification from Chapter 5, as follows:

$$\begin{aligned}
 \Theta^{val} = (& \\
 & G^{val} = \left(\begin{array}{l} as, on \end{array} \right), \\
 & \Phi_S^{val} = \left(\begin{array}{l} \phi_S^{as} := true, \phi_S^{on} := true \end{array} \right), \\
 & \Phi_C^{val}(s) = \left(\begin{array}{ll} \phi_C^{as}(s) & := (TO(s, id) \geq TO) \\ \phi_C^{on}(s) & := (QS(s) > 0) \end{array} \right) \\
 &).
 \end{aligned} \tag{9.7}$$

This PM strategies allows for the two global power states asleep and on (as assumed in Section 9.2.1). There are *no* global-level restrictions (satisfiers) for the servers to switch power states. The server-specific constraints to put a server to sleep $\phi_C^{as}(s)$ is given by comparison of the currently observed idle time-out $TO(s, id)$ to exceed the time-out threshold variable TO that has been set for the design. When the job dispatcher assigns jobs to a sleeping server, it needs to be woken, which is taken care of with the constraint $\phi_C^{on}(s)$, that wakes a server when the currently observed queue of that server $QS(s)$ has more than 0 jobs.

Example 9.4. Design $d = (5, 10, random)$ represents a job dispatcher and PM strategy, for which three simple variables have been set, in which: (i) a new server is turned on when the queue sizes of all currently running servers is greater than 5, (ii) servers shut down after 10 seconds of inactivity, and (iii) non-determinism is resolved via a random selection. $\square$

9.4 Two Implementations

This section shows the implementation of the power and performance high-level model in two different ways to enable the automatic evaluation of many job dispatching policies for load balancing and compare the results. Section 9.4.1 provides the DACSIM implementation and configuration, whereas Section 9.4.2 describes the IDSL implementation and configuration (as proposed in [B5]).

9.4.1 The DACSIM Implementation

In Chapter 4–6, a simulation framework has been proposed that allows for the analysis of power and performance trade-offs for data centres that save energy via PM. Below, the configuration of the data centre simulation framework, and an extension for description and implementation of job dispatching policies in an environment with PM features is elaborated.

Job and Data Centre Configuration

The DACSIM framework is configured according to the system description (of Section 9.2). The basic load balancer and its environment are implemented with one agent for the load balancer and one agent for each server. The load balancer distributes the workload by injecting jobs to the servers. These jobs are injected in simple queues inside the server agents.

Table 9.5–9.6 show all the parameters used. Jobs arrive at the dispatcher according to a Poisson process with rate λ (job/s). The service time of each job

Parameter	R_{pc}	R_{id}	R_{as}	R_{sl}	R_{wk}
Assigned Value	200 W	200 W	14 W	200 W	200 W

Table 9.5: DACSIM configuration **power** parameters

Parameter	λ	μ	α_{sl}	α_{wk}
Assigned Value	exp(1.0)	det(1.0)	det(10.0)	det(10.0)

Table 9.6: DACSIM configuration **performance** parameters

at each server is deterministic, i.e., the server finishes jobs in a fixed amount of time ($\frac{1}{\mu}$ seconds). The power model is configured to support four power states. The awareness of power states in the models allows to compute power consumption by rewarding each power state with a power consumption. Note that state “processing” is the only power state in which jobs are served. Furthermore, the PM module (as introduced in Chapter 5) has been enabled and configured with the PM strategy Θ^{val} , as defined in (9.7).

Simulator Configuration

The mean power consumption $E[P]$ and mean response times $E[R]$ are computed using the batch means method. The batch means method requires the length of the simulation (t_{sim}) to be very long, which is selected to be 100 000 model seconds, and the system should be stable after some warm-up (wup) period, which is selected to be 500 model seconds.

Extension for Job Dispatching Policies

Recall that the load balancer injects jobs in the queues of agents of the servers. Therefore, the load balancer needs a policy to determine where each job should go. The policies, introduced in Chapter 4 have only two simple options: (i) random and (ii) shortest queue next. So, the load balancer required an extension to support policies (cf. Section 9.4.1).

In order to implement the new policies, information is required about the size of the queue of each server and about the current power state. In order to select a server that support these policies an expression should be defined to

reward each server. The extension consist of a module class that has access to all the relevant information, such that it can rate the servers based on the three parameters (cf. Section 9.3.2). Additionally, a parameter variation experiment of the framework is implemented that allows for parallel computation of the averages of many designs.

9.4.2 The iDSL Implementation

In this subsection, we describe the essentials of the iDSL implementation. iDSL [21] is a formal language and solution chain to evaluate the performance of service-oriented systems. iDSL supports both simulation and model checking, via a transformation to the MODEST TOOLSET [75], as means to evaluate large numbers of complex designs. Finally, iDSL presents its predictions intuitively via visualisations and understandable (aggregated) metrics.

In essence, iDSL allows to evaluate conceptual models for service-oriented systems by means of a language. For iDSL to support load balancers, a language has been defined that contains (i) a policy, (ii) job characteristics, and (iii) data centre configuration (e.g., power consumption per state and transition times). A more extensive description of iDSL and how the load balancer implementation operates can be found in [B5].

9.5 Experimental Results

We show the results of the evaluation by discussing the power and performance effects, that many job dispatching policies for load balancing, has (in Section 9.5.1) and discuss their validity by comparison of the results obtained from both implementations (in Section 9.5.2).

9.5.1 Power-Performance Trade-off

Figures 9.3-9.4 show the DACSIM results for all designs, where each dot represents one job dispatching policy with different values for the idle time-out $TO \in \{1, 2, 3, 4, 5, 7.5, 10, 15, 30\}$ and the queue size threshold $q \in \{1, 2, 3, 5, 7, 10, 15, 20, 30, 40\}$. The only difference between the two figures is the colouring based on these policy values TO and q . In Figure 9.3, designs with values for TO are marked with the colour blue; the lightness of blue gradually decreases when the value of TO increase. Figure 9.4 has similar blue colours with a gradually decreasing lightness when the value of q increases.

Figure 9.3: Average power consumption and response times for many designs in DACSIM grouped by **time-out** TO ranging from the lower (light blue) to the higher values (dark blue) of TO .

Figure 9.4: Average power consumption and response times for many designs in DACSIM grouped by **queue size threshold** q ranging from the lower (light blue) to the higher values (dark blue) of q .

Figure 9.5: Comparison of the iDSL (on the y -axis) on DACSIM (on the x -axis) results for many designs, using **random** non-determinism elimination.

Figure 9.6: Comparison of the iDSL (on the y -axis) on DACSIM (on the x -axis) results for many designs, using **fixed order** non-determinism elimination.

From both figures, a power-performance trade-off is found, i.e., multiple designs exist which all are considered to approach the optimal values. The near optimal values found are depicted as the designs closest to the origin, which seem to approach an invisible curve referred to as the Pareto optimal frontier. The interesting fact about our chosen job dispatching policy is the ability to optimise and control the power and performance trade-off through only the values of TO and q .

In Figure 9.3, we observe that, for designs, that increasing the time-out threshold TO in the job dispatching policy, their position in the curve moves closer to the Pareto optimal frontier. Despite that larger values of TO seem to dominate the Pareto optimal frontier, it should be noted that the larger values of TO at some point will be more concentrated at the upper left part of the plot, i.e., it moves to the situation where TO is really large, meaning *no* servers are put to sleep, such that the overall power consumption is maximum ($E[P] \approx 4 \cdot 200 \text{ W} = 800 \text{ W}$) and performance is unhindered by machines sleeping and waking up ($E[R] \approx 1 \text{ s}$).

Figure 9.4 plots again all designs, but now with a blue gradient for the queue size threshold q . If the time-out threshold values TO are *not* chosen too large, increasing the queue size threshold q of the job dispatching policy allows us to determine how much power is traded in for performance. Note that, the resolution of any remaining non-determinism for the job dispatching policy is left out as it did *not* show any noteworthy differences.

9.5.2 Validity of the Outcomes

We assess the validity of the IDSL and DACSIM approaches by comparing their performance outcomes for many different designs. The following distance measure, which returns the ratio differences, is used to compare outcomes:

$$\delta(v_1, v_2) = \max\left(\frac{v_1}{v_2}, \frac{v_2}{v_1}\right) - 1 \quad (9.8)$$

The measure is partly like a metric, viz., $\delta(v, v) = 0$, $\delta(v_1, v_2) = \delta(v_2, v_1)$, and $\delta(av_1, av_2) = \delta(v_1, v_2)$. However, the triangular property $\delta(v_1, v_2) + \delta(v_2, v_3) \geq \delta(v_1, v_3)$ does not hold.

Figure 9.5 and 9.6 show the experimental outcomes of IDSL (on the y -axis) and DACSIM (on the x -axis) for resolving non-determinism with the random (in Figure 9.5) or the fixed order (in Figure 9.6) way, respectively. Note that the distance δ is visualised around the diagonal for values 0, 0.1, 0.2 and 0.3. Generally, the results of both implementations match, because most designs are

located near the diagonal. For half of all the designs, both the average response times of IDSL and DACSIM differed less than 6%. For 80% of the designs, this is less than 13%.

9.6 Conclusions

In this chapter, a model has been developed to evaluate the performance and energy consumption of load balancers. In this model, a powerful policy language has been defined that decides to which server jobs are assigned by observation of the system variables, e.g., queue sizes of servers.

In order to evaluate the performance and energy trade-off of many policies, we have implemented two load balancers with exactly the same specifications in IDSL and in DACSIM.

Evaluation of many policies shows that parameter q , the queue threshold for switching servers on, is useful to resolve the power-performance trade-off. We observed that the parameter TO , the idling time of servers before put to sleep, allows to *moves towards* the so-called Pareto optimal frontier, i.e., both the efficiency of power and performance improve. The parameter q , the queue size at which sleeping servers are woken, determines the *position on* this Pareto optimal frontier, i.e., it determines how much power is traded in for performance. We also showed parameter q is only effective if the value of TO is small enough to enable the trade-off and large enough to be optimal.

For cross-model validation, the evaluated results of both implementations have been compared. We found that the majority of the designs show similar results. To improve the value of the validation, more designs could be compared and an extension towards comparison of power metrics could be made. The cross-model validation shows evidence that a similar (simplified) data centre with a job dispatcher for load balancing have been modelled and analysed by the two implementations using noticeably different modelling and analysis approaches.

Experimental Micro Data Centres

In this chapter, we experimentally investigate the possibility of using cheap, low power and widely supported hardware in the form of a micro data centre. In order to do so, we decided to investigate a well-known application area that copes with large chunks of data. For this purpose, we present an experimental setup consisting of multiple RASPBERRY PI 2 MODEL Bs (RPi2s) that has been used to build a fully functional cluster, that allows for distributed processing of large amounts of distributed data. We experimentally validated the proposed setup and compared it with a real data centre cluster by analysis of its performance, scalability, energy consumption, temperature and manageability. We propose a high concurrency and low power setup in a single data centre rack unit (1U) form factor as an interesting alternative to traditional rack servers. Additionally, we validate the simulation models in DACSIM with our experimental setup by calibrating the models and comparison of the measured and analytical results.

This chapter is organised, as follows. First, an introduction is provided in Section 10.1. Section 10.2 elaborates an well-known solution to cope with large chunks of data. Then, our experimental micro data centre setup is presented equipped with this solution in Section 10.3. Our experimental setup is thoroughly tested with cluster-specific benchmarks in Section 10.4, and application-specific performance benchmarks in Section 10.5. The case study and validation are elaborated in Section 10.6. The conclusions are described in Section 10.7.

This work is based on the publication [B6], and served in bachelor publications from five students supervised by us.

10.1 Introduction

In data centres, the density of servers has increased significantly in the past years [37]. New technologies emerge, e.g., blade servers, that not only decrease the physical appearance of what used to be an entire rack full of servers, but also decrease power consumption by implementing new technologies. One of these relatively new technologies is the ARM processor, which could actually fit well with the increasing demand for modularity in data centres. RASPBERRY PIs (RPis) are small single-board computers with ARM processors onboard, that require very little energy. Since these RPis can operate as fully functional servers, these should be considered as a serious alternative.

This poses the challenge to create a realistic server cluster with RPis for use in data centres. For this purpose, we consider a frequently used data centre application that is concerned with the processing of large chunks of data considered too large for traditional data processing applications, i.e., we consider a solution in the so-called field of the *big data*. The main task concerning big data is to query large chunks of data to retrieve valuable information. In order to do so, a big data solution often requires a distributed way of data processing and distributed storage of these large amounts of data.

In this chapter, we explore the capabilities of RPis for micro data centres, thereby focussing on benchmarks and measurements of power, performance, temperature and hardware allocation of an experimental setup with RPi cluster that has been designed to fit within a data centre. These different aspects allow us to analyse three main design criteria of a flexible future proof data centre [9, p.6], namely: scalability, performance and manageability.

Several other cluster projects exist for the RASPBERRY PI MODEL B(+) (RPi), which we briefly highlight below for comparison to our project.

- The *Iridis-pi* cluster with 64 RPis was built by Cox et al. [42]. A Message Passing Interface was used to communicate between the Raspberry Pis. The research was done to investigate what the performance of a low-power high performance cluster was. It was designed as a portable and passively cooled cluster for educational purposes.
- Tso et al. [174] built a data centre consisting of 56 RPis that offers a cloud computing testbed including virtualisation management tools called the *Glasgow Raspberry Pi Cloud*. It was built for practical research on cloud computing without the limitations of simulation.
- Kiepert [93] created a *Beowulf* cluster for a PhD assignment. It was built for collaboratively processing sensor data in a wireless sensor network.

The Raspberry Pi cluster offers an alternative in case of the main cluster is unavailable.

- The *Bolzano Raspberry Pi* cluster consists of 300 RPis and was made as an affordable energy-efficient computer cluster by Abrahamsson et al [2]. Applications such as a green research testbed and as a mobile data center are evaluated. Their main goal was to introduce a cluster of RPis on a larger scale.

The RPi clusters described above are for research, application performance and cluster mobility. They applied the first generation RPi which offers significantly lower performance than the newer second generation RPi, which we use. Our research is different in providing benchmarks of the temperature, power consumption and performance with a well-known solution for big data. Moreover, our cluster has also been developed with the specific goal to be helpful in the validation process of our power and performance simulation models (as discussed in Chapter 4).

Consequently, this chapter contributes with a small case study of a computational cluster that is used to solve the well-known mathematical problem of RSA factoring for the purpose of validation. The main purpose of the case study presented is comparison of power and performance results from the experimental RPi2 setup and from calibrated models in the simulation framework DACSIM (as introduced in Chapter 4) for validation purposes.

10.2 A Key Application for Data Centres

This section provides background on a solution for big data, which is currently one of the key applications in data centres. *Big data* concerns the data that is too large and complex to be processed by traditional data processing applications. Below, we elaborate a solution that allows us to distribute processing of large chunks of data distributed over multiple storage devices.

APACHE HADOOP [168] is an open source framework which offers necessary components for the distributed processing of large amounts of distributed data, using simple programming models like map/reduce. It has been designed to scale well from one to thousands of machines. HADOOP offers high-availability options for detecting and recovering from failures in both hardware and software. HADOOP is used for applications like risk modelling and recommendation engines which have petabytes of data to be analysed.

Map/reduce [47] as implemented in HADOOP is a programming model to allow for simple distributed processing of large data sets. A map/reduce

program consists of two steps. The map step performs filtering and sorting. The reduce step can then do further computations on the output of the maps, which is usually a summarizing operation. Depending on the program the map and/or reduce tasks can be parallelised.

In HADOOP 2, *Yet Another Resource Negotiator* (YARN) [169] was introduced as a new resource management layer. YARN handles workload management and monitoring, manages high availability features and allows for more programming models next to just map/reduce.

Big server manufacturers like DELL, HP and SUPERMICRO offer all kinds of servers for HADOOP applications. HADOOP usually runs on a multitude of rack servers, that are often allocated to a single rack unit (1U) and contain eight or more storage drives. Whereas these 1U rack servers are relatively cheap, it brings a lot of space and energy overhead when scaling the number of rack servers. If compared to the more expensive, yet higher density alternative, the so-called blade servers. Blade servers house vertically placed blades combined with a single power supply and network access for all blades combined, which makes them more space efficient [108] allowing for about 50% more servers than with the traditional 1U rack servers. A powerful feature of HADOOP is that it is fully functional with just a single server, yet at the same time scalable to thousands of servers.

10.3 System Description

In this section, the software and experimental setup are elaborated. First a short summary is given of the device used in the micro data centre, in Section 10.3.1. In the experimental setup our own micro data centre is elaborated for HADOOP in Section 10.3.2. The HADOOP software used in our experimental setup for distributed processing is discussed in Section 10.3.3. Additionally, the possibility to install our proposed experimental setup in an actual data centre form factor are discussed in Section 10.3.4.

10.3.1 RASPBERRY PI 2

The RASPBERRY PI 2 MODEL B [152] (as depicted in Figure 10.1) is a small, cheap yet feature packed single-board computer. It is based on the BROADCOM BCM2836 system on a chip which offers a 900MHz quad-core ARMv7 CPU combined with 1 GB of RAM and could be bought for about \$35 in the year 2015. Detailed specifications can be found in Table 10.2. 16GB ADATA PREMIER PRO UHS-I microSD cards are used as the storage solution.

10.3.2 Experimental Setup

A total number of eight RPi2s is used in our experimental setup. A setup diagram for HADOOP is displayed in Figure 10.3. The numbers in the setup diagram correspond to the physical setup shown in Figure 10.4-10.5. The number (2) indicates a small router/switch that is connected to the power supply. The number (1) shows the eight RPi2s. For HADOOP there is one master node and at most seven slave nodes.

DIETPI [96] is used as the operating system for the individual nodes. It is a lightweight version of RASPBIAN which is the Linux distribution specifically tailored for the RPi2 to prevent overhead from background processes.

10.3.3 Software Setup

A basic HADOOP [168] installation is a composition of multiple components. For the purpose of high throughput for large data sets, streaming capabilities and high fault-tolerance HADOOP is equipped with a distributed file system for HADOOP (HDFS). As a consequence, several processes are added for that purpose. The *NameNode* is the main process which keeps track of where all files are distributed and replicated. It is the main access point for all clients and processes and runs on the master node. The *SecondaryNameNode* keeps a

Hardware Type	Hardware Specification
System on Chip	BROADCOM BCM2836
Ethernet	Onboard 10/100 ETHERNET RJ45 jack
USB	Four USB 2.0
Video out	HDMI 1.4
Audio	2 × analog
CPU	900MHZ QUAD-CORE ARM CORTEX-A7
GPU	DUAL-CORE VIDEOCORE IV MULTIMEDIA CO-PROCESSOR
Card slot	Micro SD

Figure 10.1: RASPBERRY PI 2 MODEL B. [152]

Figure 10.2: RASPBERRY PI 2 MODEL B hardware specifications [152].

Figure 10.3: Hadoop design.

Figure 10.4: Project setup with stacked RPIs in a 3D printed rack.

Figure 10.5: Project setup close-up with an indication of the (1) application nodes, and (2) network connectivity (cf. Figure 10.3).

recent backup of the *NameNode* so the *NameNode* can be restored if it might go down. The *DataNode* processes run on the remaining slave nodes and handle data storage and retrieval.

A second component used in the HADOOP software is the resource management software APACHE YARN [169] for the purpose of consistent delivery of operations, security, and data governance tools in the HADOOP clusters. YARN consists of a *ResourceManager*, which manages all jobs in the system, and on each slave node a *NodeManager*. The *NodeManager* process handles the execution of jobs allocated to a slave node. Finally, the *JobHistoryServer* keeps track of all completed jobs and their logs.

A natively compiled version of HADOOP 2.6.0 with YARN was configured

Figure 10.6: Vertical and tilted RPi2 in a 1U server.

Figure 10.7: Proposed RPi2 rack layout.

in conjunction with ORACLE JAVA 7 for ARM [141]. Because there are only eight available RPi2s allowing for at most eight nodes, a single master node runs the *NameNode*, *SecondaryNameNode*, *ResourceManager* and the *JobHistoryServer*. The other (scalable) (at most seven) nodes act as slaves and each runs a *NodeManager* and a *DataNode*.

The setup has 91 GB of distributed storage capacity available with a replication factor of two, which resulted in an effective amount of roughly 45 GB. YARN is concerned with assignment of resource requests (e.g. processors, memory, etc.) within the cluster. Processing capacity is assigned to each application by allocating a so-called *container slots*. YARN has been configured in such a way that two container slots can run concurrently on a single slave node. This gives 14 available container slots for HADOOP to allocate tasks to in the experimental setup.

10.3.4 Cluster in Data Centre Server Racks

For the RPi2 to be usable in an enterprise environment, it must fit in standardised server racks. Hardware breaks all the time in data centres, so it should be easily accessible and replaceable to keep the data centre manageable. One disadvantage of current RPi2 is the placement of the power connector and Ethernet connector. The connectors are placed perpendicular to each other which makes it harder to place the boards in a confined space. In order to keep the manageability of the data centre, two designs are proposed.

The rack must contain a power supply with sufficient ports and power to handle all RPi2. The casing must contain some fans to generate airflow.

Standard data centre racks contain often 42U of space. As defined by the

EIA-310 standard a single U is 44.50 mm high [86]. The dimensions of a 1U rack is defined as 450 mm wide, 44.43 mm high and at most 739.775 mm deep [87]. The RPi2 is 85.60 mm wide, 56 mm deep and 21 mm high. It has four standard mounting holes for screws or spacers to fit through.

The most efficient way to place the RPi2 in a small contained space is with the power connector facing downwards. So it can be connected to power on the bottom of the rack, and to Ethernet on the side, which would allow the easiest access to a RPi2. Unfortunately, as can be seen in Figure 10.6, a vertically placed RPi2 is a little higher than a standard U, so a bigger 1.5U rack should be used to make it fit. A variation can be tried by tilting the RPi2 boards so they fit in a 1U rack. The effect of this approach is shown in Figure 10.6. Because of the required low angle, practically no overlap between the RPi2s can exist. This removes the main advantage of this approach.

The most obvious way to place the boards is to stack them in pairs of two and fill up the rack. Stacks can easily be secured on the bottom of the rack server by using spacers. The downside to this approach is the accessibility of the RPi2, as either the top one or both RPi2 have to be removed. 12 RPi2 fit next to each other in the rack, this gives 24 boards for a single row. While keeping space for all cables and connectors, four rows fit in the width of a rack server. With the power supply the estimated amount is 72 RPi2 for a 1U rack, seen in Figure 10.7.

In order to provide all boards with Ethernet a 2U switch will be needed as a 1U switch can house a maximum of 48 Ethernet ports.

10.4 Cluster Benchmarking

This section elaborates benchmarks and measurements on power, temperature, storage, memory and network to test the cluster as if in a data centre environment.

10.4.1 Storage and Memory Performance

For basic system benchmarks, the SYSBENCH suite [98] has been used. It serves as a tool to quickly determine system performance without setting up any complex software.

The SD card storage was tested by running random and sequential storage tests. An test data amount of 4 GB was prepared with SYSBENCH. The benchmarks were run with a maximum execution time of 300 seconds. The memory test sequentially read and wrote 512 MB of data to memory.

Benchmark	Transfer Speed (in MB/s)
Random storage read	9.9718
Random storage write	1.2604
Random storage read/write	3.4046
Sequential storage read	17.7400
Sequential storage write	6.3972
Sequential storage rewrite	13.0000
Sequential memory read	207.5000
Sequential memory write	177.0200

Table 10.1: Transfer speed for the SD storage and RPi2 memory benchmarks computed with the SYSBENCH suite.

Table 10.1 shows that the write performance of the SD cards is low. The read performance of the SD card is below expectations, which promised 40 MB/s for sequential read operations but achieved barely half that speed. The RPi2's memory is sequentially read at 207 MB/s and written at 177 MB.

10.4.2 Energy Consumption

The energy consumption is measured with a simple setup. A prototyping PCB with two USB connectors and jumper wires are used to allow for a multimeter (ELRO M990) to connect for voltage and current measurements of a single RPi2. This way the actual power usage of the RPi2 is measured, because the energy-(in)efficiency of the power supply is *not* taken into account, i.e., measurements done at the wall outlet are expected to have a higher power usage.

The power consumption was measured under several workloads to find out what effect different kind of operations have on the power consumption of the RPi2. SYSBENCH is used to consistently stress different parts of the board.

The RPi2 has a power consumption of at most 2.1 W in this test as shown in Table 10.2. A normal server needs about 500 W [121], so 238 RPi2s take as much power on one server.

Benchmark	Current (in A)	Voltage (in V)	Power (in W)
CPU 1 core	0.340	4.84	1.65
CPU 2 cores	0.365	4.79	1.75
CPU 3 cores	0.392	4.77	1.87
CPU 4 cores	0.415	4.78	1.99
Memory test	0.440	4.79	2.11
Storage read	0.442	4.77	2.11
Storage write	0.395	4.77	1.89
Idle	0.315	4.78	1.51

Table 10.2: Power consumption observed using a multimeter for several benchmarks of a single RPi2.

10.4.3 Network Performance

I_{PERF}3 [57] was used to find out whether the network, the storage or the memory is a bottleneck by reading/writing from/to the different mediums [57]. The RPi2 uses a 12.5 MB/s Ethernet connection which is connected via a combined USB 2.0/Ethernet chip. This is important as HADOOP shuffles a large amount of data around the network, video streaming needs to transport a lot of data to the user and in between the servers. To find out if there is a bottleneck, 60 second I_{PERF}3 benchmarks with TCP congestion windows of 133 KB have been executed from memory to memory, memory to SD storage and from SD storage to memory.

For every throughput benchmarks the TCP congestion window is 133 KB. From the results in Table 10.3 the write performance of the RASPBERRY PI 2

Write direction	Avg. T (in MB/s)
memory → memory	11.7
memory → SD storage	3.04
SD storage → memory	11.8

Table 10.3: The average throughput (Avg. T) for different write direction between the memory and SD storage computed with the I_{PERF}3 benchmark.

memory and/or the SD card forms a bottleneck with only a throughput of about 3 MB/s. This number is in line with the results from the SYSBENCH write tests which were between 1.26 MB/s and 6.4 MB/s for random and sequential writes respectively. USB 2.0/Ethernet causes some overhead, therefore it has only a throughput of around 11.8 MB/s instead of the full 12.5 MB/s.

10.4.4 Temperatures

CPU temperature measurements were taken under SYSBENCH CPU stressing with different numbers of threads. During this benchmark the temperature is measured by logging the operating systems data on temperatures with a shell script. The temperature is measured on the CPU. Results are shown in Figure 10.8. The room temperature during this benchmark was around 23 °C. The cooldown phase, that occurs after the benchmark has finished, is shown in Figure 10.9. The room temperature during the cooldown phase was around 21 °C and has been measured during a separate benchmark run. By default, the RPi2 is a passively cooled board without any heat sink or fan.

When running a four-thread CPU benchmark the maximum temperature is 60 °C and the temperature is 42 °C when idle, see Figure 10.8. In Figure 10.8, after a short period of time, temperatures converge to an upper bound. After benchmark completion, the CPU cools down quickly to idle temperature, as

Figure 10.8: CPU temperature benchmark.

Figure 10.9: CPU temperature cooldown.

can be seen in Figure 10.9. Data centres often require an average room temperature of around 26 °C. In order to achieve such temperatures, additional energy is consumed for cooling. The most common workload for HADOOP and video streaming would be two CPU threads for which the temperature stays around 50 °C. So, if multiple closely-spaced RPi2s are used, some cooling is expected to be required in order to keep them working at optimal performance temperature.

10.5 Application-Specific Benchmarks

In this section, several HADOOP benchmarks are analysed to show that in a data centre environment the proposed setup has acceptable performance.

A selection of HADOOP benchmarks is made to cover the most important aspects of a HADOOP cluster. The benchmarks are taken from the HiBENCH benchmark suite [83] and from the standard HADOOP test suite. These benchmarks cover CPU bound computation and generic computation on distributed big data. A comparison is made with the CTIT cluster of the UNIVERSITY OF TWENTE where HADOOP runs on 32 DELL R415 servers.

10.5.1 The TERASORT benchmark

TERASORT (from the HiBENCH benchmark suite) is a benchmark which measures sort speed on large distributed files. The benchmark consists of a map/reduce job which creates and sorts a multiple of 100 byte rows and validates the results. A replication factor of one for the output files was forced instead of the cluster default settings. By doing so, the map/reduce performance is *not* affected by the replication of data.

Table 10.4 shows an overview of the results obtained from the TERASORT benchmark for different configurations and comparison between the RPi2 cluster and the CTIT cluster. An important first observation is that the total execution times using the CTIT clusters (#2.1) are about ten ($\approx \frac{230}{22}$) times lower compared to the RPi2 cluster (#1.1b) when sorting 1 GB of data. The average map task also took roughly ten ($\approx \frac{72}{7}$) times longer on the RPi2 cluster. The runs with more data were a lot slower on the RPi2 cluster because insufficient container slots were available in the RPi2 cluster. The average map in those cases (#1.3b vs. #2.3) took 24 times longer on the RPi2 cluster when sorting 10 GB of data. This higher ratio could be the result of the low write speed to the SD card when more data has to be handled. It should be noted that despite the lower

Experiment Identifier	RASPBERRY PI 2						CTIT		
	#1.1a	#1.1b	#1.2a	#1.2b	#1.3a	#1.3b	#2.1	#2.2	#2.3
Configuration									
Nodes	5	8	5	8	5	8	32	32	32
Slots	8	14	8	14	8	14	–	–	–
Maps	16	16	64	64	80	80	16	64	80
Reduces	8	8	8	8	8	8	8	8	8
Data (in GB)	1	1	7	7	10	10	1	7	10
Measurements									
Total execution time (in s)	366	230	3584	1747	–	3041	22	49	67
Avg. map (in s)	70	72	144	141	–	261	7	10	11
Avg. shuffle (in s)	70	88	–	698	–	830	4	19	24
Avg. reduce (in s)	48	49	1741	406	–	550	2	15	21
Estimates									
Energy consumed (in Wh)	1.0	1.0	10	8.2	–	14	28	62	84

Table 10.4: Overview of the TERASORT benchmark.

performance of the RPi2 cluster compared to the CTIT cluster, the power consumption of the RPi2 cluster (ranging between idle power consumption $1.51 \text{ W} \cdot 5/8 \text{ RPi2s} \approx 7.55 \text{ W}/12.08 \text{ W}$ and peak load power consumption $2.11 \text{ W} \cdot 5/8 \text{ RPi2s} \approx 10.55 \text{ W}/16.88 \text{ W}$, cf. Table 10.2) is expected to be much lower than the CTIT cluster (about 0.141 kW idle and 0.229 kW peak load power consumption for the DELL POWEREDGE R415 [50], such that between $0.141 \text{ kW} \cdot 32 \text{ servers} \approx 4.5 \text{ kW}$ and $0.229 \text{ kW} \cdot 32 \text{ servers} \approx 7.3 \text{ kW}$ power is consumed).

For each experiment, we have provided an indication of the estimated energy consumed (in Ws) by multiplying the power consumed with the total execution time. For all experiments (#1.1a–#2.3), the energy consumption ranges between the idle and peak load energy consumption. For the RPi2 cluster the peak load energy consumption is expected to be a more accurate estimate considering its low capacity. For the CTIT cluster, on the other hand, the idle energy consumption is expected to be a more accurate estimate considering its high capacity. Consequently, only an indication of the peak energy consumption of the RPi2 cluster and the idle energy consumption of the CTIT cluster is provided in the table. The RPi2 cluster experiments (#1.1b, #1.2b and #1.3b) are estimated to consume respectively roughly $28 (\approx \frac{28}{1})$, $8 (\approx \frac{62}{8.2})$ and $6 (\approx \frac{84}{14})$ times less energy than the CTIT cluster experiments (#2.1–#2.3).

It should be noted that the CTIT cluster (#2.1–#2.3 in Table 10.4) has far more container slots and nodes than the RPi2 cluster. During the execution of the benchmark, enough slots were used to allocate all map/reduces at once in the CTIT cluster. Since higher number of slots deliver similar performance, the number of used slots has been left out of the table for the CTIT cluster.

An inherent problem to a smaller cluster showed up in the 7 GB run on five nodes (#1.2a in Table 10.4) and is caused by one of HADOOPs optimizations for bigger clusters. When a map task finishes on a node, HADOOP starts a reduce task on that same node since the necessary data is already there. The nodes are configured to run two concurrent tasks. With seven nodes available, this gives a total of 14 container slots of which one is the *Application Master*. With more map tasks than the amount of available container slots, part of the tasks will run sequentially. The problem is that as soon as the first batch of map tasks finishes, reduce tasks get started on the nodes, so only few container slots are available for the relatively high amount of map tasks to be completed. The reduce tasks will have a lot of idle time, because input data from the map tasks becomes available at a low pace. Adding more nodes would solve this problem as enough slots should be available to allocate the map jobs. This would bring the total running time closer to the average map time.

Since the 7 GB run allocated 64 map tasks (#1.2b in Table 10.4), it took a total of 1747 seconds to complete all jobs. The average reduce time is high, because the reducers were still waiting for new input data. The shuffle time is the time to get the required data as output by a map task to the correct reducer. As there are usually many more map tasks than there are reduce tasks, this is a vital number for fast HADOOP operations. The reducers were able to retrieve data from other nodes with a reported speed of about 11 MB/s. This means HADOOP is most of the time writing into memory, as IPERF3 showed that the write speed to the SD card is much lower over the network.

The last problem showed up for the first time when TERASORT ran with 10GB of data on 5 nodes (#1.3a in Table 10.4). If HADOOP assigns two reduce tasks to a single node, they have a lot of data to process, so the reduce tasks will use too much memory when writing their results to HDFS causing the DATANODE process to crash and get kicked out of memory causing the reduce task to fail. HADOOP may then decide to start two copies of the same job to the cluster. This amplifies the problem with a small cluster, making the chance that two are running on a single node significantly higher. This problem can likely be solved by changing the YARN configuration so that only one reduce task may run on a single node.

10.5.2 The Pi benchmark

The Pi benchmark [170] (a workload example from the HADOOP test suite) distributes the workload for the computation of the exact digits of the mathematical constant π using quasi-Monte Carlo sampling. In order to do so, it creates map tasks for computation and gathers the results with a reducer. This is typically a task that is mostly CPU intensive as compared to the network and storage requirements of this benchmark.

In Table 10.5, the Pi benchmark are shown for a setup of 5 nodes with 8 container slots (#3.1–#3.2a) and for a setup of 8 nodes with 14 container slots (#3.2b). The number π is computed in the benchmark with 10^9 samples per map. Increasing the maps or samples for the benchmark makes the estimation of π more accurate. From results from the Pi benchmarks, it became clear that the average shuffle time depends on the availability of the data for the reducers (#3.1–#3.2b). The Pi benchmark generates very small intermediate data which, if all maps can be allocated (#3.1 and #3.2b), takes only 13 seconds of shuffle time which is mostly overhead time from HADOOP due to hard coded polling intervals. The runs with 8 available container slots (#3.1–#3.2a) show

Experiment Identifier	RASPBERRY PI 2			CTIT	
	#3.1	#3.2a	#3.2b	#4.1	#4.2
Configuration					
Nodes	5	5	8	–	–
Slots	8	8	14	–	–
Maps	6	12	12	6	12
Measurements					
Total execution time (in s)	996	1975	996	40	40
Avg. map (in s)	976	981	975	32	32
Avg. shuffle (in s)	13	978	13	3	3
Avg. reduce (s)	2	2	2	0	0
Estimates					
Energy consumed (in Wh)	2.9	5.8	4.7	50	50

Table 10.5: Pi benchmark for computation of the number π .

the impact of a setup with fewer available slots than there are maps to be run, compared with 6 maps (#3.1) and 12 maps (#3.2a) with enough available nodes, the total duration depends on the speed with which individual maps are finished. The results (#3.1 and #3.2b) show that the amount of maps does *not* influence running time for the PI benchmark if enough container slots are available. Thus, we can directly compare the results between the two systems. The CTIT cluster (#4.1 and #4.2) took 40 seconds to complete the benchmark with an average map time of 32 seconds. In comparison the RPi2 cluster (#3.1 and #3.2b) took 996 seconds to complete with an average map time of 975 seconds. This means that for this CPU bound benchmark the processing cores in the CTIT cluster are roughly 25 ($\approx \frac{996}{40}$) times faster than the processing cores from the RPi2. Similar to the TERASORT benchmark, an indication of the energy consumed for each experiment is provided in the table based on the idle power consumption of the CTIT cluster and the peak load power consumption of the RPi2 cluster. The RPi2 cluster is estimated to consume roughly eight ($\approx \frac{50}{5.8}$) times less energy than the CTIT cluster.

10.6 A Small Case Study for Model Validation

In this section, we elaborate a small case study for the purpose of model validation. In order to do so, we elaborate the well scalable computation case study of the factoring of RSA keys in Section 10.6.1. We then discuss the experimental setup with RPi2 clusters, similar to the setup shown in Section 10.6.2, for yet another application, viz., RSA factoring. We configure and calibrate the simulation models with measurements and characteristics that are often easy to obtain by traditional data centre in Section 10.6.3, e.g., product specification often provides power consumption of individual servers. To validate our simulation models, we perform a comparison between power and performance estimates obtained from the simulation models and the measurements obtained from the experimental setup in Section 10.6.4.

Figure 10.10 provides an overview of how the measurements from our experimental micro data centre for the RSA factoring case study are used to calibrate and, subsequently, validate the simulation models from Chapter 4 with the same job and data centre configuration.

10.6.1 Case Study: RSA Factoring

The factoring of RSA keys [155] is a realistic example of a mathematical problem with possibilities for parallel computation. RSA factoring is a computa-

Figure 10.10: An overview of the validation approach of a micro data centre with the RSA factoring case study.

tional task that can be done in parallel while the task requires only little other resources, e.g., memory, disk space and bandwidth. The computational power offered by RPi2s can be used to perform such parallel computational tasks.

RSA is used for many security applications such as X509 web certificates [40] and SSH [190]. RSA factoring concerns the cracking of the RSA keys by finding two large primes p and q , while only n (that is, the product of the unknowns p and q) is known. While the key generation procedure involves more steps this is sufficient to crack RSA keys, because cracking the RSA keys is considered to be as hard as factoring [161] (although quantum computing might change this [160]). For RSA factoring, a naive trial division algorithm for integer factoring [188] is used, because it is our purpose to only have a computational-intensive task. So, RSA factoring serves as a basic and scalable example of well-known mathematical problem that requires a significant amount of computational time.

10.6.2 Experimental Setup

For this experiment, a total of 9 RPi2 nodes have been used. One of the nodes has been configured as the master node that distributes the work uniformly at random to its slave nodes. Each RPi2 has a total number of 4 cores, such that on

each RPi2 (including the master node), 4 workers (slave nodes) are available. This gives a total of 36 available workers.

Concerning the job characteristics, exactly 1000 public keys with a fixed length of 63 bits are generated and inserted at the start of the measurement.

Performance measurements, e.g., computation times, are recorded on the master node in a database. The power consumption measurements are done with the BRENNENSTUHL PM 231E energy meter [84], which has a standard deviation of ± 0.2 W or ± 1 % according to the manufacturer.

Below, we provide the measurements that are used to configure and calibrate the power and performance simulation models. Subsequently, the measured results obtained for comparison with the simulated results for model validation purposes are elaborated. We categorise the various measurements in (i) measurements for calibration, which are observations that are often easy to obtain in traditional data centres, and (ii) measurements for validation, which are often time-consuming to obtain in traditional data centres.

Calibration Measurements A single worker computes a key between 40–200 s with an average computation time of 46.68 s. The measured average power consumption for the power supply with 5 RPi2s is 8.1 W idle and 13.1 W processing and for the power supply with 4 RPi2s; it is 6.5 W idle and 10.7 W processing. The network switch consumes a fixed 1.8 W of power.

Validation Measurements The completion of factoring 1000 RSA keys took a total computational time of 1298.7 s. An average power consumption of 25.405 W is measured for the experimental setup.

10.6.3 Simulated Micro Data Centre

In order to configure the simulation framework DACSIM, several easy to obtain power and performance parameters are required.

For the *performance* models of the simulator, the service rates of the simulation models are set to follow an exponential distribution with mean $\mu = 1/11.67 = 0.086$ keys/s, because one RPi2 computes one job every $46.68/4 = 11.67$ s. Exactly 1000 jobs are injected to the dispatcher at time $t = 0$ of the simulation and distributed in uniformly at random.

For the *power* models of the simulator, the processing power consumption reward R_{pc} is set to 2.644 W and the idle power consumption reward R_{id} is set to 1.622 W. The idle power consumption is computed by $(8.1+6.5)/9 \approx 1.622$ W and the processing power consumption is computed by $(13.1 + 10.7)/9 \approx 2.644$ W. A fixed surplus of 1.8 W is added to total power consumption via the cascading model of the simulator.

10.6.4 Model to Measurements Comparison

The completion of factoring 1000 keys for the simulation took 1317.5 s, whereas the experiment took a total computational time of 1298.7 s (a difference of 1.4%). The simulations predicted an average micro data centre power consumption of 25.595 W, whereas the experiment measured an average micro data centre power consumption of 25.405 W (a difference of 0.7%).

The minor differences are likely to be ascribed to power supply efficiency, power meter accuracy, and possible differences in the probability distribution of the job service times. The experiment took the full 1298.7 s to be executed, whereas the simulation took only a fraction of a second to be executed. The comparison illustrates the potential of our simulation models to provide useful estimates with easy to obtain information in only a small amount of time, and the potential of an experimental micro data centre to assist in model validation.

10.7 Conclusions

The contribution of this chapter is a fully functional distributed HADOOP setup with acceptable performance in the form of a micro data centre consisting of multiple RASPBERRY PI 2 MODEL B (RPi2)s. A high concurrency and low power setup that fits in a small 1U standardised form factor is proposed. This cheap setup is especially beneficial when lower performance is acceptable compared to expensive performance clusters. In the case of our two applications, acceptable performance is indeed attained, which is shown with the aid of several application-specific benchmarks. Moreover, several benchmarks are performed on the cluster to ensure it functions properly inside data centre. A network benchmark confirms an acceptable performance by showing that both applications approach the maximum network bandwidth of about 11.8 MB/s under full load. An amount of 72 RPi2s in a 1U rack is expected to result in a highly concurrent rack with acceptable performance and that consumes with a power usage of only 10's of watts. In comparison to the CTIT cluster that easily consumes kilowatts of power, programs with bigger map/reduce jobs like the TERASORT benchmark ran only 24 times slower than this cluster and the PI benchmark ran only 25 times slower than this cluster. Moreover, the expected energy consumption of the CTIT cluster is 6 to 28 times higher for the TERASORT benchmark and 8 times higher for the PI benchmark than this cluster. These numbers are promising when realising that the RPi2s have *not* yet an optimised architecture for support of a gigabit connection over USB and improved SD card reader performance. Before scaling this setup in a data

centre environment, an appropriate solution to the large number of cables is still required for manageability purposes. Moreover, a comparison between measurements from the micro version of the data centre and the simulation framework DACSIM shows similar estimates for power and performance with a small case study of a computational-intensive RSA factoring algorithm. This comparison shows that this laboratory setup can be used as a good alternative to the costly and risky, and thus less accessible, data centre experiments for model validation purposes.

Conclusions

This chapter provides a summary of all relevant findings and overall conclusions of this dissertation. Moreover, answers are provided to the research questions as stated in Chapter 1. Additionally, we discuss recommendations for future research.

This chapter is organised, as follows. First, a summary of the findings is provided in Section 11.1. In Section 11.2, answers to the research questions are elaborated. Section 11.3 provides possible directions for future research.

11.1 Summary

In this dissertation, power and performance have been analysed in the context of energy-efficient data centres through the use of models. The reason for addressing this subject has been motivated by the magnitude and significant increase at which energy is consumed by data centres worldwide, which has many implications for and led to global attention in society, industry and science (as shown in Chapter 1). At the same time, data centres have to be designed in such a way that acceptable performance is achieved. While many energy saving techniques have already been significantly improving the energy-efficiency with still acceptable performance, the magnitude of the worldwide energy consumption still remains and motivates further investigation. Therefore, we propose model-based analysis as a useful tool to increase practical insights in the highly complex infrastructure of data centres.

We have focussed our research on the two major contributors to reducing energy in data centres, viz., (i) advanced cooling strategies and (ii) power proportionality through advanced *power management* (PM) and load balancing. Since many other research addresses the problem of energy-efficient data centre with model-based analysis, we have distinguished ourselves from the literature by introducing a way to analyse both power and performance at the same time in our models. This allows us to analyse the power-performance trade-offs that are often caused by energy saving techniques. Reducing the challenges to minimisation problems of power within the boundaries of acceptable performance, hide the fact that a power-performance trade-off might exist in some cases. Awareness and, if possible, control of these trade-offs provide substantial amount of flexibility for data centre owners regarding design decisions. Moreover, we explored the potential of saving energy by combining the strength of advanced cooling control with that of advanced PM strategies.

Our main goal has been to acquire meaningful insights in both the power and performance of data centres to assist energy saving techniques using model-based analysis (cf. Section 1.3). In order to achieve this goal, we defined five research questions (RQ.1–RQ.5). In RQ.1, we looked for two sets of models, one for a numerical and one for a simulation modelling approach, that allows us to analyse power and performance of data centres within the context of advanced PM. These models allowed us to search for a way to structurally specify and evaluate PM strategies, in RQ.2, that is helpful for determine the quality of multiple PM strategies. In RQ.3, we focussed on a combination of PM strategies with advanced cooling strategies using the obtained results from RQ.1 and RQ.2. In the search for model validation and insights in data centres, a micro

data centre setup is investigated in RQ.4. In RQ.5, we are looking for ways to calibrate and validate the models obtained from RQ.1 to ensure and improve quality of the models.

11.2 Revisiting Research Questions

This section provides answers to the research questions proposed in Chapter 1. The first research question was formulated as follows:

Research Question 1

How can analysis of numerical and simulation models assist in obtaining useful insight saving energy in data centres while maintaining good performance?

Since models have been very helpful to analyse many other systems, we have been looking for ways to map existing modelling approaches and techniques on the energy-efficiency problem of data centres, in Chapter 3 and Chapter 4. In this way, we obtained insights in a safe experimental environment with low experimental costs.

We proposed the use of *Stochastic Petri net* (SPN) models to describe single and multiple server data centre models with capabilities for advanced PM. These SPN models were used to describe relevant power and performance measures, that allowed us to obtain insight in *power-performance trade-offs*. We have shown that the control of PM efficiently allows us to reduce power consumption at the cost of a higher mean response time. The advantages of analysis of our SPN models are the *rapid* computation through numerically solving the underlying Markov chain and the *intuitive* nature of reasonably high-level SPN models. On the other hand, this approach has some limiting assumptions with regard to the job characteristics and heterogeneity of the servers.

For this reason, we have developed another set of models suitable for power-performance analysis of energy-efficient data centre through *discrete-event simulation*. We provided a mixture of agent-based models built from underlying queueing models, state-chart models and functions for analysis that resulted in the simulation framework DACSIM. Moreover, we included a linear model for infrastructural components that also consume energy in a data centre due to a cascade effect. The mixture of models and the resulting framework allowed us to obtain quick insights in both *transient* and *steady-state* behaviour of data centres with heterogeneous servers for any possible workload. With

insights in typical metrics like power consumption, response times and power state utilisation, we compared data centres with different workload, design and configured with several example advanced PM strategies. Computation is relatively slower with discrete-event simulation compared to the numerical solution, however, the model assumptions are less restrictive and models are easier to extend in the mature simulation modelling tool ANYLOGIC.

The next research question is related to determining the quality of advanced PM strategies for data centres, which was formulated as follows:

Research Question 2

How can power management strategies be structurally specified and evaluated for the purpose of saving energy in data centres with performance constraints using models obtained in the context of RQ.1?

We showed that advanced PM related choices, e.g., wake up or put a server to sleep, often depend on observable quantities from sensors and logging in data centres. We noticed that much of the literature covered already many PM strategies that often follow the same principles. Since policies for such PM related choices are endless, we have defined an *unambiguous* and *structural language* to describe PM strategies in Chapter 5. Our language allowed us to describe a wide variety of PM strategies that allows us to decide the behaviour of PM based on the observable quantities. With this language, we have described PM strategies inspired by the literature and extensions thereof, thereby enabling well-structured analysis of these PM strategies for power and performance metrics through an implementation in DACSIM.

In the literature, we often found metrics that were used to provide an indication of the quality of the proposed PM strategies. We extended this towards a study dedicated to compare the quality of underlying PM strategy principles. In order to do so, we have described a method for evaluation of these PM strategies in Chapter 6. We characterised the PM strategies with four qualities categories, viz., (i) *efficiency*, (ii) *stability*, (iii) *robustness* and (iv) *adaptability*, and defined metrics that gave expression to each quality. We compared the PM strategies inspired by the literature and extensions thereof to a base case by implementation and analysis of all metrics in DACSIM. We observed that compared to the base case these PM strategies (for a given job and data centre configuration), the energy-efficiency increased, while the other qualities are maintained as well.

Recall that a major contributor to saving energy are advanced cooling strategies, the next research question focusses on the potential of combining such advanced cooling and PM control, which was formulated as follows:

Research Question 3

How can power management strategies be combined in the wider context of thermal-aware models for the purpose of saving even more energy in data centres using the models obtained in the context of RQ.1?

To answer this question, the existing power and performance models for simulation as proposed in Chapter 4 were extended by integration of models that allow for advanced cooling strategies in Chapter 7. We integrated existing thermodynamical models that take many physical parameters of the cooling infrastructure into account. Existing control strategies that affect the supply temperature and distribution of jobs are both integrated as well. In order to integrate the models, we added a *hierarchy* to the performance model from Chapter 4 that allows for the cooperation with the thermodynamical models. Furthermore, the optimal job distribution suggested by the advanced cooling control was interpreted in terms of allocation of jobs to a data centre rack. Also, a more *refined estimate* of the energy consumed by the cooling was provided, using the cascade model. The simulation framework DACSIM has been extended with an independent module that supports these thermal-aware models.

For us to obtain meaningful insights in the effects on power, performance and thermodynamics of each energy saving technique, we studied four distinct cases, which were: a data centre with (i) basic functionality, (ii) advanced cooling, (iii) advanced PM; and (iv) a combination thereof. While individual energy saving techniques gave improvements, we found that advanced cooling strategies in combination with PM strategies gave the best energy savings with a balanced temperature for all units inside a rack and without SLA violations. This showed that there is good potential for combinations of energy saving techniques in a data centre.

The next research question discusses the usefulness of our proposed set of models by means of validation and cross-model comparisons, which is formulated as follows:

Research Question 4

How to calibrate and validate the set of models obtained in the context of RQ.1?

The two different sets of models, from Chapter 3 and Chapter 4, required calibration and validation to show their value. Many of the parameters of our models were calibrated with values based on empirical studies found in the literature. We perform a cross-model validation, in Chapter 4, between the simulation models and the models solved numerically, and showed similar results for the power and performance estimates. We perform another cross-model validation between the simulation framework and an implementation with similar assumption in a general purpose model checker. We show for many load balancing variations that the observed response times and power consumption match for these two models with a difference of less than 13% for 80% of all configurations.

We modelled data centre workloads using measurements of a real data centre to calibrate our models with realistic workload assumptions in Chapter 8. We showed for a data trace of 20 million entries/jobs over two times a period of about 21 days that the service time distribution can be well approximated with a mixture of normal distributions by a new fitting algorithm. Also, we showed that jobs arrive, as assumed, according to a Poisson process. We show that relevant differences are observed in the mean power consumption when calibrating the service time of simulation models with a mixture of normal distributions compared to the traditional exponentially distribution of the service time with equal means.

An environment that would cost less and is less risky than real data centre experiments would be very beneficial for model validation and the creation of useful insights, which is formulated in the following research question:

Research Question 5

What can be the role of a micro data centre set up in providing useful insights in data centres? How do these compare to the model-based approaches?

To answer this question, we proposed and developed an experimental micro data centre setup consisting of multiple RASPBERRY PI 2 MODEL B's in

Chapter 10. We performed power, performance and temperature measurements using a variety of benchmarks. While the setup is relatively cheap, we show its versatility with a fully functional distributed HADOOP setup that is often used for handling big data. To make it an actual micro data centre, we showed how the setup would fit in a data centre infrastructure. We compare with those of a server cluster in a data centre.

Moreover, measurements from the micro data centre are compared to the simulation framework with a small case study addressing a well scalable and computational-intensive RSA factoring algorithm. The comparison shows similar results, that entail this micro data centre setup can serve as a good alternative to the costly data centre experiments for model validation purposes.

11.3 Recommendations for Future Research

In this section of the chapter, we propose possible directions for future research. We categorise these directions into modelling in Section 11.3.1, energy saving techniques in Section 11.3.2, and validation in Section 11.3.3.

11.3.1 Modelling

For both the SPN models (from Chapter 3) and the simulation models (from Chapter 4) numerous extensions could be added to support more refined models. Whereas refined models provide much more accurate results, the computation complexity often increases and much more prior knowledge is necessary to analyse the models. Therefore, finding the right level of abstraction is key. Below, we discuss directions which concern modelling and are expected to be worth investigating.

SPN Model Extensions

The proposed SPN models, from Chapter 3, could be extended towards models with (i) different PM strategies, e.g., with some form of hysteresis; (ii) dynamic PM, e.g., dynamic voltage and frequency scaling; (iii) multiple server types (speeds) and a mixtures of job sizes and inter-arrival times; (iv) sleep and hibernate states for the servers; (v) virtualisation; and (vi) thermal-aware data centres. These extensions could be achieved by repeating parts of the Petri net or an alternative related modelling formalisms with fewer limiting assumptions. The proposed SPN models still allow for the usage of efficient numerical

methods. However, future model extensions might require discrete-event simulation of these SPN models, but tools like MÖBIUS [36] and TIMENET [68] also supports this.

Simulation Model Extensions

Future extensions to the presented simulation models, from Chapter 4, are advised, among others, (i) energy-efficiency measures based on dynamic voltage and frequency scaling; (ii) large-scale data centre setting with heterogeneous servers, a mixture of job sizes and inter-arrival times; and (iii) virtualisation. Other future work includes a full validation cycle of the models with actual measurements from a data centre.

Domain-Specific Languages in DACSIM

Adding Domain Specific Languages (DSLs) in DACSIM could make the tool more accessible and easier to configure for research purposes.

Throughout the chapters many DACSIM configurations are proposed, this could be extended with a DSL for job and data centre characteristics. Currently, input is provided at the start of every simulation run. Moreover, snapshots of this input are saved in a configuration file. The language proposed in Chapter 5 could also be extended with a DSL for PM strategies. Currently, each time a new PM strategy is defined according to the specification, a new (Java) class has to be added and the code has to be recompiled before execution.

DSLs would provide a more intuitive way to describe DACSIM configuration and PM strategies. It is strongly advised to consider this recommendation when cooperating or extending research using DACSIM.

11.3.2 Energy Saving Techniques

Alternative, refined and other combinations of energy saving techniques for data centres could be further studied for more insights obtained from model-based power and performance analysis. Below, we mention directions which concern energy saving techniques and are expected to be worth investigating.

Meaningful Specification Extensions

The PM strategy specification, from Chapter 5 could be extended in many other promising research directions as proposed in the literature, such as job sched-

uling and power management based on infrastructure composition [185] and geographical location of the data centres [51].

Extended Study of Combined Energy Saving Techniques

The study from Chapter 7 could be extended towards combined controllers for all workload levels and studying different ways of combining power- and performance-aware controllers with thermal-aware controllers. Also, current analysis can be extended by studying the transient phases as a consequence of fluctuating workload conditions.

Furthermore, other energy saving techniques could be added to the analysis, e.g., the energy reduction and potential performance losses caused by performance degradation and migration with virtualisation. Also, the potential knowledge each additional model refinement for energy saving has to offer for decision making of other energy saving techniques would be a promising direction to further explore, e.g., base power management decisions on the temperatures in the racks.

11.3.3 Validation

To determine the degree to which the models correspond to reality, the models could be validated more by comparison with existing model alternatives and a more extensive validation. Below, we elaborated directions which concern validation and are expected to be worth investigating.

DACSIM Versus Other Data Centre Simulators

A well-known open-source toolkit CLOUDSIM [31] allows to simulate cloud computing scenarios and allows to specify (textually) data centre models with virtual machines, applications, users, scheduling and provisioning. This tool obtains utilisation, response times, execution times and energy consumption metrics from simulation runs. It is advised to investigate the capabilities of CLOUDSIM in comparison to our DACSIM simulation models from Chapter 4.

Many other tools like (CLOUDNETSIM++ [120], OpenDC [90], BIGHOUSE [122], DCSIM [173], GDCSIM [72], GREENCLOUD [178] and MDCSIM [112] that often use network simulators such as OMNET++ [140], NS-2 [171] and NS-3 [172]) exist that are being used to analyse data centres with different characteristics, that would be useful make comparisons with.

Full Validation Cycle

A full validation cycle of the models with calibration from actual measurements in data centres and comparison of measurements and estimates is recommended. A validation cycle consist in the first place of a calibration of all model parameters to measurements from a data centre, as done for the workload in Chapter 9. Secondly, it gives an indication of accuracy of the model estimates by comparison with the measured outcomes. In order to complete the validation cycle after model calibration in Chapter 8, a comparison has to be made between the estimated and observed power consumption.

Validation of Power Management Strategies

Additional validation of our PM strategies, from Chapter 5, three-step approach would increase the value of the work. A comparison of simulation and measured results could be considered for a realistic data centre configuration and workloads based on actual measurements in these data centres.

Improve Fitting Algorithm

The fitting algorithm, from Chapter 8, allows for various extensions, including (i) adapting the sensitivity of finding local extrema by change of parameters dynamically in the algorithm; (ii) better detecting hidden distributions; (iii) improving the refinement process; (iv) exploring alternatives to PROFIDO; and (v) compare (combinations of) other distribution types than solely normal distributions or other fitting approaches (e.g., a comparison with a mixture of phase type distributions [154]). Furthermore, the sampling for speeding up the computation of the normal distributions still needs to be studied for the amount of time saved, accuracy and effects of other statistical sampling methods instead of solely systematic sampling.

Bibliography

- [1] P. Abeles, *Efficient Java Matrix Library*, 2017. [Online]. Available: <http://ejml.org/> (cit. on p. 124).
- [2] P. Abrahamsson, S. Helmer, N. Phaphoom, L. Nicolodi, N. Preda, L. Miori, M. Angriman, J. Rikkila, X. Wang, K. Hamily and S. Bugoloni, "Affordable and Energy-Efficient Cloud Computing Clusters: The Bolzano Raspberry Pi Cloud Cluster Experiment", in *Proc. of the 5th Int. Conf. on Cloud Computing Technology and Science*, IEEE, Dec. 2013, pp. 170–175. DOI: 10.1109/CloudCom.2013.121. [Online]. Available: <http://ieeexplore.ieee.org/document/6735414/> (cit. on p. 173).
- [3] M. A. Adnan, R. Sugihara, Y. Ma and R. K. Gupta, "Energy-Optimized Dynamic Deferral of Workload for Capacity Provisioning in Data Centers", in *Proc. of the Int. Green Computing Conf.*, IEEE, Jun. 2013, pp. 1–10. DOI: 10.1109/IGCC.2013.6604515. [Online]. Available: <https://ieeexplore.ieee.org/document/6604515/> (cit. on p. 30).
- [4] M. Afman, "Energiegebruik Nederlandse Commerciële Datacenters: Nieuwbouwplannen en Ontwikkeling Energiegebruik", CE Delft, Delft, Tech. Rep., 2014, pp. 1–28 (cit. on p. 2).
- [5] American Society of Heating Refrigerating and Air Conditioning Engineers, "Data Center Power Equipment Thermal Guidelines and Best Practices", Tech. Rep., 2016, pp. 1–60. [Online]. Available: https://tc0909.ashraetcs.org/documents/ASHRAE_TC0909_Power_White_Paper_22_June_2016_REVISED.pdf (cit. on p. 21).
- [6] Anaconda Inc, *Anaconda*, 2018 (cit. on p. 136).
- [7] L. Andrew, M. Lin, A. Wierman and Z. Liu, "Algorithms for Dynamic Capacity Provisioning", in *Proc. of the 10th Int. Conf. on Optical Internet*, 2012, pp. 73–74. [Online]. Available: <https://ieeexplore.ieee.org/document/6245948/> (cit. on p. 30).
- [8] AnyLogic, *AnyLogic: Multimethod Simulation Software*, 2000. [Online]. Available: <http://www.anylogic.com/> (cit. on p. 58).

- [9] M. Arregoces and M. Portolani, *Data Center Fundamentals*. Indianapolis: Cisco Press, 2003, pp. 1–1064 (cit. on pp. 15, 19, 172).
- [10] M. Avgerinou, P. Bertoldi and L. Castellazzi, “Trends in Data Centre Energy Consumption under the European Code of Conduct for Data Centre Energy Efficiency”, *Journal of Energies*, vol. 10, no. 12, p. 1470, Sep. 2017. DOI: 10.3390/en10101470. [Online]. Available: <http://www.mdpi.com/1996-1073/10/10/1470/> (cit. on p. 2).
- [11] A. Azimzadeh and N. Tabrizi, “A Dynamic Power Management Schema for Multi-tier Data Centers”, in *Proc. of the 3rd IEEE Int. Conf. on Big Data Intelligence and Computing Technology Congress*, IEEE, Nov. 2017, pp. 1114–1118. DOI: 10.1109/DASC-PICom-DataCom-CyberSciTec.2017.180. [Online]. Available: <http://ieeexplore.ieee.org/document/8328524/> (cit. on pp. 80, 105).
- [12] P. Barford and M. Crovella, “Generating representative Web workloads for network and server performance evaluation”, *Journal of ACM SIG-METRICS Performance Evaluation Review*, vol. 26, no. 1, pp. 151–160, Jun. 1998. DOI: 10.1145/277858.277897. [Online]. Available: <http://portal.acm.org/citation.cfm?doid=277858.277897> (cit. on p. 134).
- [13] L. A. Barroso and U. Hölzle, “The Case for Energy-Proportional Computing”, *Journal of Computer*, vol. 40, no. 12, pp. 33–37, Dec. 2007. DOI: 10.1109/MC.2007.443. [Online]. Available: <http://ieeexplore.ieee.org/document/4404806/> (cit. on pp. 17, 40, 155).
- [14] —, “The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines”, *Synthesis Lectures on Computer Architecture*, vol. 8, no. 3, pp. 1–154, 2009. [Online]. Available: http://www.valleytalk.org/wp-content/uploads/2013/10/WSC_2.4_Final-Draft.pdf (cit. on p. 18).
- [15] R. Basmadjian, F. Niedermeier and H. De Meer, “Modelling and Analysing the Power Consumption of Idle Servers”, in *Proc. of the 2nd IFIP Conf. on Sustainable Internet and ICT for Sustainability*, 2012, pp. 1–9 (cit. on pp. 4, 29, 32, 55).
- [16] R. Basmadjian, S. Rainer and H. De Meer, “A Generic Methodology to Derive Empirical Power Consumption Prediction Models for Multi-Core Processors”, in *Proc. of the 3rd IEEE Int. Conf. on Cloud and Green Computing*, IEEE, 2013, pp. 167–174. DOI: 10.1109/CGC.2013.32. [Online]. Available: <http://ieeexplore.ieee.org/document/6686024/> (cit. on pp. 29, 32).

- [17] E. Bauer and R. Adams, "Cloud Computing", in *Reliability and Availability of Cloud Computing*, Hoboken, NJ, USA: John Wiley and Sons, Inc., Jul. 2012, pp. 1–15. DOI: 10.1002/9781118393994.ch1. [Online]. Available: <http://onlinelibrary.wiley.com/doi/abs/10.1002/9781118393994.ch1/> (cit. on p. 15).
- [18] F. Bause, P. Buchholz and J. Kriege, "ProFiDo - The Processes Fitting Toolkit Dortmund", in *Proc. of the 7th Int. Conf. on the Quantitative Evaluation of Systems*, IEEE, Sep. 2010, pp. 87–96. DOI: 10.1109/QEST.2010.20. [Online]. Available: <http://ieeexplore.ieee.org/document/5600401/> (cit. on pp. 134, 136).
- [19] L. Benini, A. Bogliolo and G. De Micheli, "A Survey of Design Techniques for System-Level Dynamic Power Management", *Journal of IEEE Transactions on Very Large Scale Integration Systems*, vol. 8, no. 3, pp. 299–316, Jun. 2000. DOI: 10.1109/92.845896. [Online]. Available: <http://ieeexplore.ieee.org/document/845896/> (cit. on p. 152).
- [20] F. van den Berg, "Automated Performance Evaluation of Service-Oriented Systems", PhD thesis, University of Twente, Enschede, The Netherlands, Jun. 2017. DOI: 10.3990/1.9789036543590. [Online]. Available: <http://research.utwente.nl/en/publications/automated-performance-evaluation-of-service-oriented-systems/> (cit. on pp. 12, 151, 153).
- [21] F. van den Berg, J. Hooman and B. R. Haverkort, "A Domain-Specific Language and Toolchain for Performance Evaluation Based on Measurements", in *Proc. of the 19th Int. GI/ITG Conf. on Measurement, Modeling and Evaluation of Computing Systems*, R. German, K. Hielscher and U. Krieger, Eds., ser. Lecture Notes in Computer Science, vol. 10740, Springer, Cham, 2018, pp. 295–301. DOI: 10.1007/978-3-319-74947-1_21. [Online]. Available: http://link.springer.com/10.1007/978-3-319-74947-1_21/ (cit. on p. 165).
- [22] F. van den Berg, A. Remke and B. R. Haverkort, "iDSL: Automated Performance Prediction and Analysis of Medical Imaging Systems", in *Proc. of the 12th Eur. Workshop on Computer Performance Engineering*, Springer, Cham, 2015, pp. 227–242. DOI: 10.1007/978-3-319-23267-6_15. [Online]. Available: http://link.springer.com/10.1007/978-3-319-23267-6_15/ (cit. on p. 152).

- [23] A. P. Bianzino, C. Chaudet, D. Rossi and J.-L. Rougier, "A Survey of Green Networking Research", *Journal of IEEE Communications Surveys and Tutorials*, vol. 14, no. 1, pp. 3–20, 2012. DOI: 10.1109/SURV.2011.113010.00106. [Online]. Available: <http://ieeexplore.ieee.org/document/5677351/> (cit. on p. 16).
- [24] R. Birke, L. Y. Chen and E. Smirni, "Data Centers in the Wild: A Large Performance Study", IBM Research, Tech. Rep., 2012, pp. 1–12. [Online]. Available: <http://domino.research.ibm.com/library/cyberdig.nsf/papers/0C306B31CF0D3861852579E40045F17F/> (cit. on p. 71).
- [25] G. Bolch, S. Greiner, H. de Meer and K. S. Trivedi, *Queueing Networks and Markov Chains*. New York, USA: John Wiley and Sons, Inc., Aug. 1998. DOI: 10.1002/0471200581. [Online]. Available: <http://doi.wiley.com/10.1002/0471200581/> (cit. on p. 4).
- [26] T. Bostoen, S. Mullender and Y. Berbers, "Analysis of Disk Power Management for Data-Center Storage Systems", in *Proc. of the 3rd Int. Conf. on Future Energy Systems: Where Energy, Computing and Communication Meet*, 2012, pp. 1–10. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2208830> (cit. on pp. 29, 55).
- [27] —, "Power-Reduction Techniques for Data-Center Storage Systems", *Journal of ACM Computing Surveys*, vol. 45, no. 3, 2011. [Online]. Available: <http://belllabs.be/people/tombostoen/files/2012/04/csur.pdf> (cit. on pp. 29, 55).
- [28] T. Bostoen, J. Napper, S. Mullender and Y. Berbers, "Minimizing Energy Dissipation in Content Distribution Networks using Dynamic Power Management", in *Proc. of the 3rd Int. Conf. on Cloud and Green Computing*, IEEE, 2013, pp. 203–210. DOI: 10.1109/CGC.2013.40. [Online]. Available: <http://ieeexplore.ieee.org/document/6686032/> (cit. on p. 29).
- [29] D. Bruneo, "Modeling Energy-Aware Cloud Federations with SRNs", in *Transactions on Petri Nets and Other Models of Concurrency VI*, ser. Lecture Notes in Computer Science, K. Jensen, W. M. Aalst, M. Ajmone Marsan, G. Franceschinis, J. Kleijn and L. M. Kristensen, Eds., vol. 7400, Springer, Berlin, Heidelberg, 2012, pp. 277–307. DOI: 10.1007/978-3-642-35179-2_12. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-35179-2_12/ (cit. on pp. 31, 32, 53).

- [30] D. Bruneo, A. Lhoas, F. Longo and A. Puliafito, "Analytical Evaluation of Resource Allocation Policies in Green IaaS Clouds", in *Proc. of the 3rd Int. Conf. on Cloud and Green Computing*, IEEE, 2013, pp. 84–91. DOI: 10.1109/CGC.2013.21. [Online]. Available: <http://ieeexplore.ieee.org/document/6686013/> (cit. on pp. 4, 31, 32, 53, 59).
- [31] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms", *Journal of Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, Jan. 2011. DOI: 10.1002/spe.995. [Online]. Available: <http://doi.wiley.com/10.1002/spe.995/> (cit. on pp. 4, 153, 199).
- [32] M. Calzarossa and G. Serazzi, "Workload Characterization: A Survey", *Proceedings of the IEEE*, vol. 81, no. 8, pp. 1136–1150, 1993. DOI: 10.1109/5.236191. [Online]. Available: <http://ieeexplore.ieee.org/document/236191/> (cit. on p. 134).
- [33] J. Chao, *Data Centers Continue to Proliferate While Their Energy Use Plateaus*, Jun. 2016. [Online]. Available: <http://newscenter.lbl.gov/2016/06/27/data-centers-continue-proliferate-energy-use-plateaus/> (cit. on p. 3).
- [34] E. J. Chen and W. D. Kelton, "A Procedure for Generating Batch-Means Confidence Intervals for Simulation: Checking Independence and Normality", *Journal of Simulation*, vol. 83, no. 10, pp. 683–694, Oct. 2007. DOI: 10.1177/0037549707086039. [Online]. Available: <http://journals.sagepub.com/doi/10.1177/0037549707086039/> (cit. on p. 68).
- [35] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao and F. Zhao, "Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services.", in *Proc. of the 5th USENIX Symp. on Networked Systems Design and Implementation*, 2008, pp. 337–350. [Online]. Available: http://www.usenix.org/legacy/event/nsdi08/tech/full_papers/chen/chen.pdf (cit. on pp. 29, 38, 80, 90).
- [36] G. Clark, T. Courtney, D. Daly, D. Deavours, S. Derisavi, J. M. Doyle, W. H. Sanders and P. Webster, "The Mobius Modeling Tool", in *Proc. of the 9th Int. Workshop on Petri Nets and Performance Models*, Aachen, Germany: IEEE, 2001, pp. 241–250. DOI: 10.1109/PNPM.2001.953373. [Online]. Available: <http://ieeexplore.ieee.org/document/953373/> (cit. on pp. 38, 39, 198).

- [37] J. Clark, *Raising Data Center Power Density*, 2013. [Online]. Available: <http://www.datacenterjournal.com/raising-data-center-power-density/> (visited on 26/10/2015) (cit. on p. 172).
- [38] Computerworld New Zealand, *Intel, IBM Strike Deal to Lower PC Ownership Costs*, 1996. [Online]. Available: http://www.computerworld.co.nz/article/519210/intel_ibm_strike_deal_lower_pc_ownership_costs/ (visited on 26/09/2016) (cit. on p. 82).
- [39] R. W. Conway, "Some Tactical Problems in Digital Simulation", *Journal of Management Science*, vol. 10, no. 1, pp. 47–61, Oct. 1963. DOI: 10.1287/mnsc.10.1.47. [Online]. Available: <http://pubsonline.informs.org/doi/abs/10.1287/mnsc.10.1.47/> (cit. on p. 68).
- [40] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", Tech. Rep., May 2008. DOI: 10.17487/rfc5280. [Online]. Available: <http://www.ietf.org/rfc/rfc5280.txt> (cit. on p. 187).
- [41] G. D. Costa, A. Oleksiak, W. Piatek and J. Salom, "Minimization of Costs and Energy Consumption in a Data Center by a Workload-based Capacity Management", in *Proc. of the 3rd Int. Workshop on Energy-Efficient Data Centres*, S. Klingert, M. Chinnici and M. Rey Porto, Eds., ser. Lecture Notes in Computer Science, vol. 8945, Springer, Cham, 2014, pp. 102–119. DOI: 10.1007/978-3-319-15786-3_7 (cit. on p. 32).
- [42] S. J. Cox, J. T. Cox, R. P. Boardman, S. J. Johnston, M. Scott and N. S. O'Brien, "Iridis-Pi: A Low-Cost, Compact Demonstration Cluster", *Journal of Cluster Computing*, vol. 17, no. 2, pp. 349–358, Jun. 2014. DOI: 10.1007/s10586-013-0282-7. [Online]. Available: <http://link.springer.com/10.1007/s10586-013-0282-7> (cit. on p. 172).
- [43] F. da Cruz, *Columbia Difference Tabulator*, 2013. [Online]. Available: <http://www.columbia.edu/cu/computinghistory/packard.html> (visited on 06/03/2018) (cit. on p. 14).
- [44] A. F. Cuervo, "Load Balancing in the Datacenter", in *Site Reliability Engineering: How Google Runs Production Systems*, S. Chavis, Ed., O'Reilly Media, Inc., 2016 (cit. on p. 28).
- [45] Datacenter Dynamics Intelligence, "Global Data Center Power 2013", Datacenter Dynamics, Tech. Rep., 2013, pp. 1–5. [Online]. Available: <http://www.dcd-intelligence.com/Products-Services/Open-Research/Global-Data-Center-Power-2013> (cit. on p. 2).

- [46] M. Dayarathna, Y. Wen and R. Fan, "Data Center Energy Consumption Modeling: A Survey", *Journal of IEEE Communications Surveys and Tutorials*, vol. 18, no. 1, pp. 732–794, 2016. DOI: 10.1109/COMST.2015.2481183. [Online]. Available: <http://ieeexplore.ieee.org/document/7279063/> (cit. on p. 4).
- [47] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", *Journal of Communications of the ACM*, vol. 51, no. 1, p. 107, Jan. 2008. DOI: 10.1145/1327452.1327492. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1327492> (cit. on p. 173).
- [48] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall and W. Vogels, "Dynamo: Amazon's Highly Available Key-Value Store", *Journal of ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, pp. 205–220, Oct. 2007. DOI: 10.1145/1323293.1294281. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1294281> (cit. on p. 40).
- [49] Dell, "Dell Takes Holistic Approach to Address Data Center Power and Cooling Issues", Tech. Rep., 2005. [Online]. Available: https://www.dell.com/downloads/global/products/pedge/en/data_cntr_pwr_clng.pdf (cit. on p. 16).
- [50] —, *ENERGY STAR Power and Performance Data Sheet: Dell PowerEdge R415 Featuring the Dell Energy Smart 500W PSU and AMD 4180 CPU*. [Online]. Available: http://www.dell.com/downloads/global/products/pedge/en/R415_2P_4180_500W_Energy_Star_Data_Sheet.pdf (visited on 16/08/2018) (cit. on p. 183).
- [51] X. Deng, D. Wu, J. Shen and J. He, "Eco-Aware Online Power Management and Load Scheduling for Green Cloud Datacenters", *Journal of IEEE Systems Journal*, vol. 10, no. 1, pp. 78–87, Mar. 2016. DOI: 10.1109/JSYST.2014.2344028. [Online]. Available: <http://ieeexplore.ieee.org/document/6877627/> (cit. on p. 199).
- [52] A. Al-Dulaimy, W. Itani, A. Zekri and R. Zantout, "Power Management in Virtualized Data Centers: State of the Art", *Journal of Cloud Computing*, vol. 5, no. 1, p. 6, Dec. 2016. DOI: 10.1186/s13677-016-0055-y. [Online]. Available: <https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-016-0055-y/> (cit. on p. 33).

- [53] C. Dupont, T. Schulze, G. Giuliani, A. Somov and F. Hermenier, "An Energy Aware Framework for Virtual Machine Placement in Cloud Federated Data Centres", in *Proc. of the 3rd Int. Conf. on Future Energy Systems Where Energy, Computing and Communication Meet*, New York, New York, USA: ACM Press, 2012, pp. 1–10. DOI: 10.1145/2208828.2208832. [Online]. Available: <https://dl.acm.org/citation.cfm?id=2208832> (cit. on pp. 29, 32).
- [54] C. Earnes and R. Earnes, *A Computer Perspective: Background to the Computer Age*. Cambridge, USA: Harvard University Press, 1973, p. 95 (cit. on p. 14).
- [55] R. El Abdouni Khayari, R. Sadre and B. R. Haverkort, "Fitting World-Wide Web Request Traces with the EM-Algorithm", *Journal of Performance Evaluation*, vol. 52, no. 2-3, pp. 175–191, Apr. 2003. DOI: 10.1016/S0166-5316(02)00179-7. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S0166531602001797> (cit. on pp. 134, 141).
- [56] Emerson Network Power, "Energy Logic: Reducing Data Center Energy Consumption by Creating Savings that Cascade Across Systems", *White Paper of Emerson Electric Co*, 2009 (cit. on pp. 16, 38, 58, 59, 65, 95, 111).
- [57] ESnet, *iPerf3*, 2015. [Online]. Available: <https://iperf.fr/> (cit. on p. 180).
- [58] European Commission, "Code of Conduct on Data Centres Energy Efficiency Version 1.0", Ispra, Tech. Rep., 2008, pp. 1–20. [Online]. Available: http://ec.europa.eu/information_society/activities/sustainable_growth/docs/datacenter_code-conduct.pdf (cit. on p. 19).
- [59] X. Fan, W.-D. Weber and L. A. Barroso, "Power Provisioning for a Warehouse-Sized Computer", *Journal of ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 13–23, Jun. 2007. DOI: 10.1145/1273440.1250665. [Online]. Available: <https://dl.acm.org/citation.cfm?id=1250665> (cit. on p. 29).
- [60] D. Ferrari, "Workload Characterization and Selection in Computer Performance Measurement", *Journal of IEEE Computer*, vol. 5, no. 4, pp. 18–24, 1972. DOI: 10.1109/C-M.1972.216939. [Online]. Available: <http://ieeexplore.ieee.org/document/1641573/> (cit. on p. 134).

- [61] W. Fischer and K. Meier-Hellstern, "The Markov-Modulated Poisson Process (MMPP) Cookbook", *Journal of Performance Evaluation*, vol. 18, no. 2, pp. 149–171, Sep. 1993. DOI: 10.1016/0166-5316(93)90035-S. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/016653169390035S/> (cit. on p. 134).
- [62] G. S. Fishman, "Grouping Observations in Digital Simulation", *Journal of Management Science*, vol. 24, no. 5, pp. 510–521, Jan. 1978. DOI: 10.1287/mnsc.24.5.510. [Online]. Available: <http://pubsonline.informs.org/doi/abs/10.1287/mnsc.24.5.510/> (cit. on p. 68).
- [63] A. Gandhi, "Dynamic Server Provisioning for Data Center Power Management", PhD thesis, Carnegie Mellon University, 2013, pp. 1–174 (cit. on pp. 32, 59, 80).
- [64] A. Gandhi, S. Doroudi, M. Harchol-Balter and A. Scheller-Wolf, "Exact Analysis of the M/M/k/setup Class of Markov Chains via Recursive Renewal Reward", in *Proc. of the ACM SIGMETRICS Int. Conf. on Measurement and Modeling of Computer Systems.*, vol. 41, ACM New York, NY, USA, 2013, pp. 153–166. DOI: 10.1145/2494232.2465760 (cit. on pp. 4, 30, 32, 53, 64, 125).
- [65] A. Gandhi and M. Harchol-Balter, "How Data Center Size Impacts the Effectiveness of Dynamic Power Management", in *Proc. of the 49th Annual Allerton Conf. on Communication, Control, and Computing*, IEEE, Sep. 2011, pp. 1164–1169. DOI: 10.1109/Allerton.2011.6120299. [Online]. Available: <https://ieeexplore.ieee.org/document/6120299> (cit. on pp. 4, 30, 32).
- [66] A. Gandhi, M. Harchol-Balter and M. A. Kozuch, "Are Sleep States Effective in Data Centers?", in *Proc. of the 3rd Int. Green Computing Conf.*, IEEE, Jun. 2012, pp. 1–10. DOI: 10.1109/IGCC.2012.6322260. [Online]. Available: <http://ieeexplore.ieee.org/abstract/document/6322260> (cit. on pp. 30, 32, 40, 80, 90, 155).
- [67] A. Gandhi, M. Harchol-Balter, R. Raghunathan and M. A. Kozuch, "AutoScale: Dynamic, Robust Capacity Management for Multi-Tier Data Centers", *Journal of ACM Transactions on Computer Systems*, vol. 30, no. 4, pp. 1–26, Nov. 2012. DOI: 10.1145/2382553.2382556. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2382556> (cit. on pp. 30, 32).

- [68] R. German, C. Kelling, A. Zimmermann and G. Hommel, "TimeNET: A Toolkit for Evaluating Non-Markovian Stochastic Petri Nets", *Journal of Performance Evaluation*, vol. 24, no. 1-2, pp. 69–87, Nov. 1995. DOI: 10.1016/0166-5316(95)00010-U. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/016653169500010U/> (cit. on p. 198).
- [69] R. Ghosh, V. K. Naik and K. S. Trivedi, "Power-Performance Trade-offs in IaaS Cloud: A Scalable Analytic Approach", in *Proc. of the 41st Int. Conf. on Dependable Systems and Networks Workshops*, IEEE, 2011, pp. 152–157. DOI: 10.1109/DSNW.2011.5958802. [Online]. Available: <http://ieeexplore.ieee.org/document/5958802> (cit. on pp. 4, 30, 32, 53, 59).
- [70] D. Gmach, J. Rolia, L. Cherkasova, G. Belrose, T. Turicchi and A. Kemper, "An Integrated Approach to Resource Pool Management: Policies, Efficiency and Quality Metrics", in *Int. Conf. on Dependable Systems and Networks With FTCS and DCC*, IEEE, 2008, pp. 326–335. DOI: 10.1109/DSN.2008.4630101. [Online]. Available: <http://ieeexplore.ieee.org/document/4630101/> (cit. on p. 29).
- [71] D. Gmach, J. Rolia, L. Cherkasova and A. Kemper, "Resource Pool Management: Reactive versus Proactive or Let's be Friends", [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.295.7905> (cit. on p. 29).
- [72] S. K. Gupta, R. R. Gilbert, A. Banerjee, Z. Abbasi, T. Mukherjee and G. Varsamopoulos, "GDCCSim: A Tool for Analyzing Green Data Center Design and Resource Management Techniques", in *Proc. of the 2nd Int. Green Computing Conf. and Workshops*, IEEE, Jul. 2011, pp. 1–8. DOI: 10.1109/IGCC.2011.6008612. [Online]. Available: <http://ieeexplore.ieee.org/document/6008612/> (cit. on pp. 4, 199).
- [73] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P. P. Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q. M. Malluhi, N. Tziritas, A. Vishnu, S. U. Khan and A. Zomaya, "A Survey and Taxonomy on Energy Efficient Resource Allocation Techniques for Cloud Computing Systems", *Journal of Computing*, vol. 98, no. 7, pp. 751–774, Jul. 2016. DOI: 10.1007/s00607-014-0407-8. [Online]. Available: <http://link.springer.com/10.1007/s00607-014-0407-8/> (cit. on p. 116).

- [74] A. Hartmanns and H. Hermanns, “A Modest Approach to Checking Probabilistic Timed Automata”, in *Proc. of the 6th Int. Conf. on Quantitative Evaluation of Systems*, IEEE, Sep. 2009, pp. 187–196. DOI: 10.1109/QEST.2009.41. [Online]. Available: <http://ieeexplore.ieee.org/document/5290840/> (cit. on p. 153).
- [75] —, “The Modest Toolset: An Integrated Environment for Quantitative Modelling and Verification”, in *Proc. of the 20th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, ser. Theoretical Computer Science and General Issues, vol. 8413, Springer-Verlag, 2014, pp. 593–598. DOI: 10.1007/978-3-642-54862-8_51. [Online]. Available: http://link.springer.com/10.1007/978-3-642-54862-8_51 (cit. on p. 165).
- [76] B. R. Haverkort, *Performance of Computer Communication Systems: A Model-Based Approach*. John Wiley and Sons, Inc., Nov. 1998 (cit. on pp. 4, 38, 41, 45, 62).
- [77] T. Henderson and R. Dvorak, *Blade Server vs Rack Server: Which Server Form Factor is Better for Your Data Center?*, 2007. [Online]. Available: <http://www.networkworld.com/article/2290914/data-center/blade-servers-vs-rack-servers.html> (visited on 04/12/2018) (cit. on p. 16).
- [78] N. Herbst, R. Krebs, G. Oikonomou, G. Kousiouris, A. Evangelinou, A. Iosup and S. Kounev, “Ready for Rain? A View from SPEC Research on the Future of Cloud Metrics”, SPEC Research, Tech. Rep., 2016, pp. 1–43. arXiv: 1604.03470 (cit. on p. 110).
- [79] Hewlett-Packard, Intel, Microsoft, Phoenix Technologies and Toshiba, “Advanced Configuration and Power Interface Specification”, Tech. Rep., 2011, pp. 1–922. [Online]. Available: <http://acpi.info/DOWNLOADS/ACPIspec50.pdf> (cit. on pp. 23, 25, 62).
- [80] A. Hinton, M. Kwiatkowska, G. Norman and D. Parker, “PRISM: A Tool for Automatic Verification of Probabilistic Systems”, in, 2006, pp. 441–444. DOI: 10.1007/11691372_29. [Online]. Available: http://link.springer.com/10.1007/11691372_29 (cit. on p. 153).
- [81] J. Hopcroft, R. Motwani and J. Ullman, *Automata theory, languages, and computation*, 24th ed. 2006. [Online]. Available: http://docs.google.com/viewer?url=http://www.academia.edu/download/31352670/19s_Automata_Theory.pdf (cit. on p. 62).

- [82] T. Horvath and K. Skadron, "Multi-Mode Energy Management for Multi-Tier Server Clusters", in *Proc. of the 17th Int. Conf. on Parallel Architectures and Compilation Techniques*, ACM New York, NY, USA, Oct. 2008, pp. 270–279. DOI: 10.1145/1454115.1454153. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1454153> (cit. on pp. 29, 80, 90, 96).
- [83] S. Huang, J. Huang, J. Dai, T. Xie and B. Huang, "The HiBench Benchmark Suite: Characterization of the MapReduce-Based Data Analysis", in *Proc. of the 26th Int. Conf. on Data Engineering Workshops*, IEEE, 2010, pp. 41–51. DOI: 10.1109/ICDEW.2010.5452747. [Online]. Available: <http://ieeexplore.ieee.org/document/5452747/> (cit. on p. 182).
- [84] Hugo Brennenstuhl GmbH and Co, *Primera-Line Energiemeettoestel PM 231 E*, 2018. [Online]. Available: <http://www.brennenstuhl.nl> (cit. on p. 188).
- [85] IBM, *IBM Announces Universal Management – Industry's Most Comprehensive Tools to Lower Total Cost of Ownership*, 1998. [Online]. Available: <http://www-03.ibm.com/press/us/en/pressrelease/2705.wss> (visited on 26/09/2015) (cit. on p. 82).
- [86] Innovation First inc., *19-inch Rack (EIA-310)*, 2007. [Online]. Available: <http://www.server-racks.com/eia-310.html> (visited on 13/08/2018) (cit. on p. 178).
- [87] —, *Rack Mounting Depth*, 2007. [Online]. Available: <http://www.server-racks.com/rack-mount-depth.html> (visited on 13/08/2018) (cit. on p. 178).
- [88] Intel Corporation, *Intel Ark*. [Online]. Available: <http://ark.intel.com/> (visited on 03/08/2018) (cit. on p. 154).
- [89] Intel and Microsoft, "Advanced Power Management (APM): BIOS Interface Specification", Tech. Rep., 1996, pp. 1–73 (cit. on p. 23).
- [90] A. Iosup, G. Andreadis, V. V. Beek, M. Bijman, E. V. Eyk, M. Neacsu, L. Overweel, S. Talluri, L. Versluis and M. Visser, "The OpenDC Vision: Towards Collaborative Datacenter Simulation and Exploration for Everybody", in *Proc. of the 16th Int. Sym. on Parallel and Distributed Computing*, IEEE, Jul. 2017, pp. 85–94. DOI: 10.1109/ISPD.2017.25. [Online]. Available: <http://ieeexplore.ieee.org/document/8121623/> (cit. on pp. 4, 199).

- [91] ISO, *International Organization for Standardization*, 2018. [Online]. Available: <https://www.iso.org/> (visited on 01/08/2018) (cit. on p. 19).
- [92] J. P. Katoen, T. Noll, T. Santen, D. Seifert and H. Wu, "Performance Analysis of Computing Servers using Stochastic Petri Nets and Markov Automata", RWTH Aachen University, Technical Report, 2013 (cit. on pp. 4, 31, 32, 59).
- [93] J. Kiepert, "Creating a Raspberry Pi-Based Beowulf Cluster", Boise State University, Tech. Rep., 2013, pp. 1–17 (cit. on p. 172).
- [94] J. v. Kistowski, H. Block, J. Beckett, K.-D. Lange, J. A. Arnold and S. Kounev, "Analysis of the Influences on Server Power Consumption and Energy Efficiency for CPU-Intensive Workloads", in *Proc. of the 6th ACM/SPEC Int. Conf. on Performance Engineering*, New York, New York, USA: ACM Press, Jan. 2015, pp. 223–234. DOI: 10.1145/2668930.2688057. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2668930.2688057> (cit. on p. 64).
- [95] S. Klingert, R. Basmadjian, C. Bunse, V. Georgiadou, G. Giuliani, G. Lovasz and M. Majanen, "Fit4green-Energy Aware ICT Optimization Policies", pp. 88–92, 2010 (cit. on pp. 29, 32).
- [96] D. Knight, *DietPi*, 2014. [Online]. Available: <http://dietpi.com/> (cit. on p. 175).
- [97] J. G. Koomey, "Growth in Data Center Electricity Use 2005 to 2010", Tech. Rep., 2011, pp. 1–20. [Online]. Available: http://www.missioncriticalmagazine.com/ext/resources/MC/Home/Files/PDFs/Koomey_Data_Center.pdf (cit. on p. 1).
- [98] A. Kopytov, "SysBench Benchmark Suite", *GitHub Repository*, 2017. [Online]. Available: <http://github.com/akopytov/sysbench/> (cit. on p. 178).
- [99] A. Krioukov, P. Mohan, S. Alspaugh, L. Keys, D. Culler and R. H. Katz, "NapSAC: Design and Implementation of a Power-Proportional Web Cluster", in *Proc. of the 1st ACM SIGCOMM Workshop on Green Networking*, New York, New York, USA: ACM Press, Aug. 2010, pp. 15–22. DOI: 10.1145/1851290.1851294. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1851294> (cit. on pp. 29, 80, 90).
- [100] N. Kroes, *Using ICT to Build Sustainable Cities*, Brussels, 2012. [Online]. Available: http://europa.eu/rapid/press-release_SPEECH-12-320_en.htm (cit. on p. 1).

- [101] P. J. Kuehn, "Energy Efficiency and Performance of Cloud Data Centers: Which Role Can Modeling Play?", *Proc. of the 5th Int. Work. on Energy Efficient Data Centres*, pp. 1–6, 2016. DOI: 10.1145/2940679.2940688. [Online]. Available: <http://doi.acm.org/10.1145/2940679.2940688/> (cit. on p. 80).
- [102] P. J. Kuehn and M. Mashaly, "Performance of Self-Adapting Power-Saving Algorithms for ICT Systems", in *Int. Symposium IFIP/IEEE on Integrated Network Management*, IEEE, 2013, pp. 720–723. [Online]. Available: <http://ieeexplore.ieee.org/document/6573062/> (cit. on pp. 4, 30, 32, 53, 59).
- [103] Y. Kuroda, A. Akai, T. Kato and Y. Kudo, "High-Efficiency Power Supply System for Server Machines in Data Center", in *Proc. of the Int. Conf. on High Performance Computing and Simulation*, IEEE, 2013, pp. 172–177. DOI: 10.1109/HPCSim.2013.6641410. [Online]. Available: <http://ieeexplore.ieee.org/document/6641410/> (cit. on p. 16).
- [104] K. G. Larsen, P. Pettersson and W. Yi, "Uppaal in a Nutshell", *Journal of Software Tools for Technology Transfer*, vol. 1, no. 1-2, pp. 134–152, Dec. 1997. DOI: 10.1007/s100090050010. [Online]. Available: <http://link.springer.com/10.1007/s100090050010/> (cit. on p. 153).
- [105] A. M. Law, *Simulation Modeling and Analysis*, 5th ed. McGraw-Hill, 2015, p. 804 (cit. on p. 58).
- [106] A. M. Law and J. S. Carson, "A Sequential Procedure for Determining the Length of a Steady-State Simulation", *Journal of Operations Research*, vol. 27, no. 5, pp. 1011–1025, Oct. 1979. DOI: 10.1287/opre.27.5.1011. [Online]. Available: <http://pubsonline.informs.org/doi/abs/10.1287/opre.27.5.1011> (cit. on p. 68).
- [107] A. M. Law and D. W. Kelton, *Simulation Modeling and Analysis*. McGraw-Hill Education - Europe, 2000, p. 305 (cit. on p. 141).
- [108] K. Leigh, P. Ranganathan and J. Subhlok, "General-Purpose Blade Infrastructure for Configurable System Architectures", *Journal of Distributed and Parallel Databases*, vol. 22, no. 2-3, pp. 197–198, Nov. 2007. DOI: 10.1007/s10619-007-7016-x. [Online]. Available: <http://link.springer.com/10.1007/s10619-007-7016-x/> (cit. on p. 174).

- [109] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts and S. Wolff, "A Brief History of the Internet", *Journal of ACM SIGCOMM Computer Communication Review*, vol. 39, no. 5, pp. 22–31, Oct. 2009. DOI: 10.1145/1629607.1629613. [Online]. Available: <https://dl.acm.org/citation.cfm?id=1629613> (cit. on p. 14).
- [110] J. C. Leite, D. M. Kusic and D. Mossé, "Stochastic Approximation Control of Power and Tardiness in a Three-Tier Web-Hosting Cluster", in *Proc. of the 7th Int. Conf. on Autonomic Computing*, New York, New York, USA: ACM Press, Jun. 2010, pp. 41–50. DOI: 10.1145/1809049.1809056. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1809056> (cit. on p. 29).
- [111] W. E. Leland, W. Willinger, M. S. Taqqu and D. V. Wilson, "On the Self-Similar Nature of Ethernet Traffic", *Journal of ACM SIGCOMM Computer Communication Review*, vol. 25, no. 1, pp. 202–213, Jan. 1995. DOI: 10.1145/205447.205464. [Online]. Available: <http://dl.acm.org/citation.cfm?id=205464> (cit. on p. 134).
- [112] S. H. Lim, B. Sharma, G. Nam, E. K. Kim and C. R. Das, "MDCSim: A Multi-Tier Data Center Simulation Platform", in *Proc. of IEEE Int. Conf. on Cluster Computing*, IEEE, 2009, pp. 1–9. DOI: 10.1109/CLUSTER.2009.5289159. [Online]. Available: <http://ieeexplore.ieee.org/document/5289159/> (cit. on p. 4, 199).
- [113] M. Lin, A. Wierman, L. L. H. Andrew and E. Thereska, "Online Dynamic Capacity Provisioning in Data Centers", in *Proc. of the 49th Annual Allerton Conf. on Communication, Control, and Computing*, IEEE, Sep. 2011, pp. 1159–1163. DOI: 10.1109/Allerton.2011.6120298. [Online]. Available: <https://ieeexplore.ieee.org/document/6120298/> (cit. on pp. 4, 30, 32).
- [114] —, "Dynamic Right-Sizing for Power-Proportional Data Centers", *Journal of IEEE/ACM Transactions on Networking*, vol. 21, no. 5, pp. 1378–1391, Oct. 2013. DOI: 10.1109/TNET.2012.2226216. [Online]. Available: <http://ieeexplore.ieee.org/document/6361254/> (cit. on p. 30, 32).
- [115] B. G. Lindsay, "Mixture Models: Theory, Geometry and Applications", *Journal of NSF-CBMS Regional Conference Series in Probability and Statistics*, vol. 5, pp. i–163, 1995. [Online]. Available: <http://jstor.org/stable/4153184> (cit. on p. 134).

- [116] J. D. C. Little, "A Proof for the Queuing Formula: $L = \lambda W$ ", *Journal of Operations Research*, vol. 9, no. 3, pp. 383–387, Jun. 1961. DOI: 10.1287/opre.9.3.383. [Online]. Available: <http://pubsonline.informs.org/doi/abs/10.1287/opre.9.3.383/> (cit. on p. 42).
- [117] Z. Liu and C. H. Xia, Eds., *Performance Modeling and Engineering*. Boston, MA: Springer US, 2008. DOI: 10.1007/978-0-387-79361-0. [Online]. Available: <http://link.springer.com/book/10.1007/978-0-387-79361-0/> (cit. on p. 80).
- [118] G. Lovász, F. Niedermeier and H. Meer, "Performance Tradeoffs of Energy-Aware Virtual Machine Consolidation", *Journal of Cluster Computing*, vol. 16, no. 3, pp. 481–496, May 2012. DOI: 10.1007/s10586-012-0214-y. [Online]. Available: <http://link.springer.com/10.1007/s10586-012-0214-y/> (cit. on p. 55).
- [119] T. Lu, M. Chen and L. L. Andrew, "Simple and Effective Dynamic Provisioning for Power-Proportional Data Centers", *Journal of IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1161–1171, Jun. 2013. DOI: 10.1109/TPDS.2012.241. [Online]. Available: <http://ieeexplore.ieee.org/document/6269026/> (cit. on p. 29).
- [120] A. W. Malik, K. Bilal, S. U. Malik, Z. Anwar, K. Aziz, D. Kliazovich, N. Ghani, S. U. Khan and R. Buyya, "CloudNetSim++: A GUI Based Framework for Modeling and Simulation of Data Centers in OM-NeT++", *Journal of IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 506–519, Jul. 2017. DOI: 10.1109/TSC.2015.2496164. [Online]. Available: <http://ieeexplore.ieee.org/document/7312485/> (cit. on pp. 4, 199).
- [121] D. Meisner, B. T. Gold and T. F. Wenisch, "PowerNap: Eliminating Server Idle Power", in *Proc. of the 14th Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, New York, New York, USA: ACM Press, 2009, pp. 205–216. DOI: 10.1145/1508244.1508269. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1508269> (cit. on p. 179).
- [122] D. Meisner, J. Wu and T. F. Wenisch, "BigHouse: A Simulation Infrastructure for Data Center Systems", in *Proc. of the IEEE Int. Sym. on Performance Analysis of Systems and Software*, IEEE, Apr. 2012, pp. 35–45. DOI: 10.1109/ISPASS.2012.6189204. [Online]. Available: <http://ieeexplore.ieee.org/document/6189204/> (cit. on pp. 4, 199).

- [123] D. A. Menasce and V. A. F. Almeida, *Capacity Planning for Web Services: Metrics, Models, and Methods*. Prentice Hall, 2002, pp. 1–572. [Online]. Available: <http://dl.acm.org/citation.cfm?id=560806> (cit. on p. 4).
- [124] D. A. Menasce, V. A. Almeida and L. W. Dowdy, *Capacity Planning and Performance Modeling: From Mainframes to Client-Server Systems*. PTR Prentice Hall, 1994, pp. 1–412. [Online]. Available: <http://dl.acm.org/citation.cfm?id=553354> (cit. on p. 4).
- [125] D. A. Menasc, M. N. Bennani and H. Ruan, “On the Use of Online Analytic Performance Models, in Self-Managing and Self-Organizing Computer Systems”, in, 2005, pp. 128–142. DOI: 10.1007/11428589_9. [Online]. Available: http://link.springer.com/10.1007/11428589_9/ (cit. on p. 29).
- [126] S. Mittal, “Power Management Techniques for Data Centers: A Survey”, Apr. 2014. arXiv: 1404.6681. [Online]. Available: <https://arxiv.org/abs/1404.6681> (cit. on p. 22).
- [127] V. Mohan Raj and R. Shriram, “Power Management in Virtualized Datacenter – A Survey”, *Journal of Network and Computer Applications*, vol. 69, pp. 117–133, Jul. 2016. DOI: 10.1016/j.jnca.2016.04.019. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1084804516300704/> (cit. on p. 33).
- [128] J. Moore, J. Chase, P. Ranganathan and R. Sharma, *Making Scheduling Cool: Temperature-Aware Workload Placement in Data Centers*, 2005. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1247365> (cit. on p. 121).
- [129] T. Mukherjee, A. Banerjee, G. Varsamopoulos, S. K. Gupta and S. Rungta, “Spatio-Temporal Thermal-Aware Job Scheduling to Minimize Energy Consumption in Virtualized Heterogeneous Data Centers”, *Journal of Computer Networks*, vol. 53, no. 17, pp. 2888–2904, Dec. 2009. DOI: 10.1016/j.comnet.2009.06.008. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1389128609002047/> (cit. on p. 22).
- [130] D. Narayanan, A. Donnelly and A. Rowstron, “Write Off-Loading: Practical Power Management for Enterprise Storage”, *Journal of ACM Transactions on Storage*, vol. 4, no. 3, 2008. DOI: 10.1145/1416944.1416949. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1416949> (cit. on pp. 29, 38).

- [131] NASA, *Climate Change: Vital Signs of the Planet*, 2008. [Online]. Available: <http://climate.nasa.gov/> (visited on 01/03/2018) (cit. on p. 1).
- [132] R. Nathuji, A. Kansal and A. Ghaffarkhah, "Q-Clouds: Managing Performance Interference Effects for QoS-Aware Clouds", in *Proc. of the 5th Eur. Conf. on Computer Systems*, New York, New York, USA: ACM Press, 2010, pp. 237–250. DOI: 10.1145/1755913.1755938. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1755938> (cit. on p. 29).
- [133] NEN, *Normalisatie En Normen*, 2018. [Online]. Available: <http://www.nen.nl/> (visited on 01/08/2018) (cit. on p. 19).
- [134] J. von Neumann, "Distribution of the Ratio of the Mean Square Successive Difference to the Variance", *Journal of The Annals of Mathematical Statistics*, vol. 12, no. 4, pp. 367–395, Dec. 1941. DOI: 10.1214/aoms/1177731677. [Online]. Available: <http://projecteuclid.org/euclid.aoms/1177731677/> (cit. on p. 68).
- [135] J. M. Nolan, "'An Inconvenient Truth' Increases Knowledge, Concern, and Willingness to Reduce Greenhouse Gases", *Environment and Behavior*, vol. 42, no. 5, pp. 643–658, Sep. 2010. DOI: 10.1177/0013916509357696. [Online]. Available: <http://journals.sagepub.com/doi/10.1177/0013916509357696> (cit. on p. 1).
- [136] A. L. Norberg, *Computers and Commerce: A Study of Technology and Management at Eckert-Mauchly Computer Company, Engineering Research Associates, and Remington Rand, 1946–1957*. MIT Press, Dec. 2005, pp. 1–384. [Online]. Available: <http://mitpress.mit.edu/books/computers-and-commerce/> (cit. on p. 14).
- [137] NumPy, *Scientific Computing Package for Python*. [Online]. Available: <http://www.numpy.org/> (cit. on p. 136).
- [138] G. O'Regan, *A Brief History of Computing*. London: Springer London, 2012. DOI: 10.1007/978-1-4471-2359-0. [Online]. Available: <http://link.springer.com/10.1007/978-1-4471-2359-0> (cit. on p. 14).
- [139] D. Ohara, "Sustainable Computing: Is It Time to Turn Off Your Servers?", *TechNet Magazine*, 2008. [Online]. Available: <http://technet.microsoft.com/en-us/magazine/cc700341.aspx> (cit. on p. 38).
- [140] OpenSim Ltd., *OMNeT++ Discrete Event Simulator*. [Online]. Available: <http://www.omnetpp.org/> (cit. on pp. 4, 199).

- [141] Oracle, *Java 7 for ARM v6/v7 Hard Float*, 2015. [Online]. Available: <http://www.oracle.com/technetwork/java/javase/downloads/jdk7-arm-downloads-2187468.html> (cit. on p. 177).
- [142] Pandas, *Python Data Analysis Library*. [Online]. Available: <http://pandas.pydata.org/> (cit. on p. 136).
- [143] C. D. Patel and A. J. Shah, “Cost Model for Planning, Development and Operation of a Data Center”, Hewlett-Packard Development Company, Tech. Rep., 2005, pp. 1–36. [Online]. Available: <http://www.hpl.hp.com/techreports/2005/HPL-2005-107R1.pdf> (cit. on p. 18).
- [144] V. Petrucci, E. V. Carrera, O. Loques, J. C. Leite and D. Mossé, “Optimized Management of Power and Performance for Virtualized Heterogeneous Server Clusters”, in *Proc. of the 11th IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing*, IEEE, May 2011, pp. 23–32. DOI: 10.1109/CCGrid.2011.15. [Online]. Available: <http://ieeexplore.ieee.org/document/5948593/> (cit. on p. 153).
- [145] E. Pinheiro, R. Bianchini, E. V. Carrera and T. Heath, “Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems”, in *Proc. of the 1st Workshop on Compilers and Operating Systems for Low Power*, vol. 180, 2001, pp. 182–195. [Online]. Available: <http://www.research.cs.rutgers.edu/~edpin/TR-440.pdf> (cit. on p. 153).
- [146] B. F. Postema, *DaCSim: A Data Centre Simulation Framework*, 2013. [Online]. Available: <http://github.com/bjornpostema/DaCSim/> (cit. on p. 69).
- [147] —, “Data Centre Workload Analysis, GitHub Repository”, *GitHub Repository*, 2017. [Online]. Available: <http://github.com/bjornpostema/datacentre-workload-analysis/> (cit. on pp. 136, 137, 143).
- [148] J. Presper Eckert and J. W. Mauchly, *Electronic Numerical Integrator And Computer*, Jun. 1947. [Online]. Available: <http://patents.google.com/patent/US3120606/> (cit. on p. 14).
- [149] Project Jupyter, *Jupyter Notebook*. [Online]. Available: <http://jupyter.org/> (cit. on pp. 134, 136).
- [150] PyMix, *The Python Mixture Package*, 2013. [Online]. Available: <http://www.pymix.org/pymix/> (cit. on p. 141).
- [151] N. Rasmussen, “High-Efficiency AC Power Distribution for Data Centers”, *White Paper of Schneider Electric*, 2007. [Online]. Available: http://www.apc.com/salestools/nran-6cn8pk/nran-6cn8pk_r2_en.pdf (cit. on p. 17).

- [152] Raspberry Pi Foundation, *Raspberry Pi 2 Model B*, 2015. [Online]. Available: <http://www.raspberrypi.org/products/raspberry-pi-2-model-b/> (visited on 26/10/2015) (cit. on pp. 174, 175).
- [153] P. Reinecke, T. Krauß and K. Wolter, "Cluster-Based Fitting of Phase-Type Distributions to Empirical Data", *Journal of Computers and Mathematics with Applications*, vol. 64, no. 12, pp. 3840–3851, Dec. 2012. DOI: 10.1016/j.camwa.2012.03.016. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0898122112002210/> (cit. on p. 134).
- [154] A. Riska, V. Diev and E. Smirni, "An EM-Based Technique for Approximating Long-Tailed Data Sets with PH Distributions", *Journal of Performance Evaluation*, vol. 55, no. 1-2, pp. 147–164, Jan. 2004. DOI: 10.1016/S0166-5316(03)00101-9. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0166531603001019/> (cit. on p. 200).
- [155] R. L. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Journal of Communications of the ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978. DOI: 10.1145/359340.359342. [Online]. Available: <http://dl.acm.org/citation.cfm?id=359342> (cit. on p. 186).
- [156] W. H. Sanders and J. F. Meyer, "Stochastic Activity Networks: Formal Definitions and Concepts", in *Lectures on Formal Methods and Performance Analysis. EEF School 2000*. Ser. Lecture Notes in Computer Science, B. Ed, H. Holger and K. Joost-Pieter, Eds., vol. 2090, Springer, Berlin, Heidelberg, 2001, pp. 315–343. DOI: 10.1007/3-540-44667-2_9. [Online]. Available: http://link.springer.com/10.1007/3-540-44667-2_9/ (cit. on p. 38).
- [157] R. Schutt and C. O'Neil, *Doing Data Science: Straight Talk from the Frontline*. O'Reilly Media, Inc., 2013 (cit. on pp. 134, 135).
- [158] SciPi, *Scientific Computing Tools for Python*. [Online]. Available: <http://www.scipy.org/> (cit. on p. 136).
- [159] A. Shehabi, S. Smith, D. Sartor, R. Brown, M. Herrlin, J. Koomey, E. Masanet, N. Horner, I. Azevedo and W. Lintner, "United States Data Center Energy Usage Report", Tech. Rep., 2016. [Online]. Available: https://eta.lbl.gov/sites/all/files/publications/lbnl-1005775_v2.pdf (cit. on pp. 2, 3, 17, 21).

- [160] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", *Journal of SIAM Review*, vol. 41, no. 2, pp. 303–332, Jan. 1999. DOI: 10.1137/S0036144598347011. [Online]. Available: <http://epubs.siam.org/doi/10.1137/S0036144598347011> (cit. on p. 187).
- [161] N. P. Smart, "The 'Naive' RSA Algorithm", in *Cryptography Made Simple*, 2016, pp. 295–311. DOI: 10.1007/978-3-319-21936-3_15. [Online]. Available: http://link.springer.com/10.1007/978-3-319-21936-3_15/ (cit. on p. 187).
- [162] A. Stefanek, R. A. Hayden and J. T. Bradley, "Fluid Analysis of Energy Consumption using Rewards in Massively Parallel Markov Models", *Proc. of the 2nd ACM/SPEC Int. Conf. on Performance Engineering*, pp. 121–132, 2011. DOI: 10.1145/1958746.1958767. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1958767> (cit. on p. 59).
- [163] A. Stevenson, *Data Centre*, 2015. [Online]. Available: http://en.oxforddictionaries.com/definition/data_centre (cit. on p. 14).
- [164] Q. Tang, S. K. S. Gupta and G. Varsamopoulos, "Energy-Efficient Thermal-Aware Task Scheduling for Homogeneous High-Performance Computing Data Centers: A Cyber-Physical Approach", *Journal of IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 11, pp. 1458–1472, Nov. 2008. DOI: 10.1109/TPDS.2008.111. [Online]. Available: <http://ieeexplore.ieee.org/document/4553706/> (cit. on p. 22).
- [165] Q. Tang, S. S. Gupta, D. Stanzione and P. Cayton, "Thermal-Aware Task Scheduling to Minimize Energy Usage of Blade Server Based Datacenters", in *Proc. of the 2nd IEEE Int. Sym. on Dependable, Autonomic and Secure Computing*, IEEE, 2006, pp. 195–202. DOI: 10.1109/DASC.2006.47. [Online]. Available: <http://ieeexplore.ieee.org/document/4030883/> (cit. on p. 125).
- [166] Tebodin Netherlands B.V., "Energemonitor ICT 2008", Ministerie van Economische Zaken, Landbouw en Innovatie, Den Haag, Tech. Rep., 2009, pp. 1–25. [Online]. Available: <http://anzdoc.com/energemonitor-ict-2008.html> (cit. on p. 2).
- [167] Technische Universität Dortmund, *ProFiDo*, 2014. [Online]. Available: http://ls4-www.cs.tu-dortmund.de/cms/de/forschung/projekte/dfg-maps/software_dfg/profido/ (cit. on p. 136).
- [168] The Apache Software Foundation, *Apache Hadoop*, 2015. [Online]. Available: <http://hadoop.apache.org/> (cit. on pp. 173, 175).

- [169] —, *Apache Hadoop YARN*, 2015. [Online]. Available: <https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html> (cit. on pp. 174, 176).
- [170] —, *Package Org.Apache.Hadoop.Examples.Pi*, 2017. [Online]. Available: <http://hadoop.apache.org/docs/r3.0.0/api/org/apache/hadoop/examples/pi/package-summary.html> (cit. on p. 185).
- [171] *The Network Simulator - ns-2*. [Online]. Available: <http://www.isi.edu/nsnam/ns/> (cit. on pp. 4, 199).
- [172] *The Network Simulator - ns-3*. [Online]. Available: <https://www.nsnam.org/> (cit. on pp. 4, 199).
- [173] M. Tighe, G. Keller, M. Bauer and H. Lutfiyya, “DCSim : A Data Centre Simulation Tool for Evaluating Dynamic Virtualized Resource Management”, *Proc. of the 8th Int. Conf. on Network and Service Management, and Workshop on Systems Virtualization Management*, pp. 385–392, 2012. [Online]. Available: <https://ieeexplore.ieee.org/document/6380046/> (cit. on pp. 4, 199).
- [174] F. P. Tso, D. R. White, S. Jouet, J. Singer and D. P. Pezaros, “The Glasgow Raspberry Pi Cloud: A Scale Model for Cloud Computing Infrastructures”, in *Proc. of the 33rd Int. Conf. on Distributed Computing Systems Workshops*, IEEE, Jul. 2013, pp. 108–112. DOI: 10.1109/ICDCSW.2013.25. [Online]. Available: <http://ieeexplore.ieee.org/document/6679872/> (cit. on p. 172).
- [175] United States Environmental Protection Agency, *Sources of Greenhouse Gas Emissions*, 2017. [Online]. Available: <https://www.epa.gov/ghgemissions/sources-greenhouse-gas-emissions/> (visited on 01/03/2018) (cit. on p. 1).
- [176] University of Groningen, *Center for Information Technology*. [Online]. Available: <http://www.rug.nl/society-business/centre-for-information-technology/> (visited on 03/08/2018) (cit. on p. 154).
- [177] —, *Peregrine - CIT HPC Projects*. [Online]. Available: <http://redmine.hpc.rug.nl/redmine/projects/peregrine/> (visited on 03/08/2018) (cit. on p. 154).
- [178] University of Luxembourg, *GreenCloud - The green cloud simulator*, 2010. [Online]. Available: <https://greencloud.gforge.uni.lu/> (cit. on pp. 4, 199).

- [179] B. Urgaonkar, P. Shenoy, A. Chandra and P. Goyal, "Dynamic Provisioning of Multi-tier Internet Applications", in *Proc. of the 2nd Int. Conf. on Autonomic Computing*, IEEE, 2005, pp. 217–228. DOI: 10.1109/ICAC.2005.27. [Online]. Available: <http://ieeexplore.ieee.org/document/1498066/> (cit. on pp. 29, 86).
- [180] Z. Usmani and S. Singh, "A Survey of Virtual Machine Placement Techniques in a Cloud Data Center", in *Proc. of Physics Procedia*, vol. 78, Elsevier, Jan. 2016, pp. 491–498. DOI: 10.1016/j.procs.2016.02.093. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1877050916000958> (cit. on p. 26).
- [181] T. Van Damme, C. De Persis and P. Tesi, "Optimized Thermal-Aware Job Scheduling and Control of Data Centers", *Journal of IFAC-PapersOnLine*, vol. 50, no. 1, pp. 8244–8249, Jul. 2017. DOI: 10.1016/j.ifacol.2017.08.1393. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S2405896317319353> (cit. on pp. 116, 119, 122, 123).
- [182] H. N. Van, F. D. Tran and J.-M. Menaud, "Performance and Power Management for Cloud Infrastructures", in *2010 IEEE 3rd International Conference on Cloud Computing*, IEEE, Jul. 2010, pp. 329–336. DOI: 10.1109/CLOUD.2010.25. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5557975> (cit. on p. 153).
- [183] A. Varasteh and M. Goudarzi, "Server Consolidation Techniques in Virtualized Data Centers: A Survey", *Journal of IEEE Systems Journal*, vol. 11, no. 2, pp. 772–783, Jun. 2017. DOI: 10.1109/JSYST.2015.2458273. [Online]. Available: <http://ieeexplore.ieee.org/document/7217792/> (cit. on pp. 26, 27).
- [184] N. Vasic, T. Scherer and W. Schott, "Thermal-aware workload scheduling for energy efficient data centers", in *Proc. of the 7th Int. Conf. on Autonomic Computing*, New York, New York, USA: ACM Press, 2010, p. 169. DOI: 10.1145/1809049.1809076. [Online]. Available: <http://portal.acm.org/citation.cfm?doid=1809049.1809076> (cit. on p. 125).
- [185] V. Villebonnet, G. Da Costa, L. Lefevre, J.-M. Pierson and P. Stolf, "Energy Aware Dynamic Provisioning for Heterogeneous Data Centers", in *Proc. of the 28th Int. Sym. on Computer Architecture and High Performance Computing (SBAC-PAD)*, IEEE, Oct. 2016, pp. 206–213. DOI: 10.1109/SBAC-PAD.2016.34. [Online]. Available: <http://ieeexplore.ieee.org/document/7789342/> (cit. on p. 199).

- [186] D. Wang, C. Ren, A. Sivasubramaniam, B. Urgaonkar and H. Fathy, "Energy storage in datacenters", in *Proc. of the 12th Int. Conf. on Measurement and Modeling of Computer Systems*, New York, New York, USA: ACM Press, 2012, p. 187. DOI: 10.1145/2254756.2254780. [Online]. Available: <http://dl.acm.org/citation.cfm?doid=2254756.2254780> (cit. on p. 17).
- [187] L. Wang, G. Von Laszewski, J. Dayal, X. He, A. J. Younge and T. R. Furlani, "Towards thermal aware workload scheduling in a data center", in *Proc. of 10th Int. Sym. on Pervasive Systems, Algorithms, and Networks*, IEEE, 2009, pp. 116–122. DOI: 10.1109/I-SPAN.2009.22. [Online]. Available: <http://ieeexplore.ieee.org/document/5381947/> (cit. on p. 22).
- [188] E. W. Weisstein, *Prime Factorization Algorithms*. (visited on 14/08/2018) (cit. on p. 187).
- [189] C. Wu and R. Buyya, *Cloud Data Centers and Cost Modeling*. Elsevier, 2015, pp. 153–155, 249–284. DOI: 10.1016/B978-0-12-801413-4.00004-0. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/B9780128014134000040> (cit. on pp. 15, 21).
- [190] T. Ylonen, "The Secure Shell (SSH) Transport Layer Protocol", Tech. Rep., Jan. 2006. DOI: 10.17487/rfc4253. [Online]. Available: <https://www.rfc-editor.org/info/rfc4253> (cit. on p. 187).
- [191] H. Zhang, S. Shao, H. Xu, H. Zou and C. Tian, "Free Cooling of Data Centers: A Review", *Journal of Renewable and Sustainable Energy Reviews*, vol. 35, pp. 171–182, Jul. 2014. DOI: 10.1016/j.rser.2014.04.017. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1364032114002445> (cit. on p. 21).
- [192] S. Zhang, Y. Liu, B. Wang and R. Zhang, "Analysis and Modeling of Dynamic Capacity Provisioning Problem for a Heterogeneous Data Center", in *Proc. of the 5th Int. Conf. on Ubiquitous and Future Networks*, IEEE, Jul. 2013, pp. 785–790. DOI: 10.1109/ICUFN.2013.6614927. [Online]. Available: <http://ieeexplore.ieee.org/document/6614927> (cit. on pp. 4, 30, 32).
- [193] W. Zhang, Y. Wen, Y. Wah Wong, K. Chuan Toh and C.-H. Chen, "Towards Joint Optimization Over ICT and Cooling Systems in Data Centre: A Survey", *Journal of IEEE Communications Surveys and Tutorials*, vol. 18, no. 3, pp. 1596–1616, 2016. DOI: 10.1109/COMST.2016.2545109. [Online]. Available: <http://ieeexplore.ieee.org/document/7438750/> (cit. on p. 116).

List of Publications

- [B1] B. F. Postema, N. J. Geuze and B. R. Haverkort, "Fitting Realistic Data Centre Workloads", in *Proc. of the 9th Int. Conf. on Future Energy Systems*, ACM Press, 2018, pp. 486–491. DOI: 10.1145/3208903.3213520. [Online]. Available: <http://dl.acm.org/citation.cfm?id=3213520> (cit. on pp. 12, 133).
- [B2] B. F. Postema and B. R. Haverkort, "Evaluation of Advanced Data Centre Power Management Strategies", in *Proc. of the 9th Int. Work. on Practical Application of Stochastic Modeling*, W. Knottenbelt, N. Thomas and M. Forshaw, Eds., vol. 337, Electronic Notes in Theoretical Computer Science, 2018, pp. 173–191. DOI: 10.1016/j.entcs.2018.03.040. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1571066118300446> (cit. on pp. 11, 101).
- [B3] B. F. Postema, T. Van Damme, C. De Persis, P. Tesi and B. R. Haverkort, "Combining Energy Saving Techniques in Data Centres using Model-Based Analysis", in *Proc. of the 4th Int. Work. on Energy-Aware Simulation and Modelling*, ACM, 2018, pp. 67–72. DOI: 10.1145/3185768.3186310. [Online]. Available: <http://dl.acm.org/citation.cfm?id=3186310> (cit. on pp. 11, 115).
- [B4] B. F. Postema and B. R. Haverkort, "Specification of Data Centre Power Management Strategies", in *Proc. of the 6th Int. Workshop on Energy-Efficient Data Centres*, 2017, pp. 284–289. DOI: 10.1145/3077839.3084025. [Online]. Available: <http://dl.acm.org/citation.cfm?id=3084025> (cit. on pp. 11, 79).
- [B5] F. van den Berg, B. F. Postema and B. R. Haverkort, "Evaluating Load Balancing Policies for Performance and Energy-Efficiency", in *Proc. of the 14th Int. Work. on Quantitative Aspects of Programming Languages and Systems*, vol. 227, 2016, pp. 98–117. DOI: 10.4204/EPTCS.227.7. [Online]. Available: <http://arxiv.org/abs/1610.08172> (cit. on pp. 12, 151, 153, 163, 165).

- [B6] N. J. Schot, P. J. Velthuis and B. F. Postema, "Capabilities of Raspberry Pi 2 for Big Data and Video Streaming Applications in Data Centres", in *Proc. of the 18th Int. GI/ITG Conf. on Measurement, Modelling and Evaluation of Dependable Computer and Communication Systems*, ser. Lecture Notes in Computer Science, vol. 9629, 2016, pp. 183–198. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-319-31559-1_15/ (cit. on pp. 12, 171).
- [B7] B. F. Postema and B. R. Haverkort, "An AnyLogic Simulation Model for Power and Performance Analysis of Data Centres", in *Proc. of the 12th Eur. Workshop on Performance Engineering*, M. Beltrán, W. Knottenbelt and J. Bradley, Eds., ser. Lecture Notes in Computer Science, vol. 9272, Springer International Publishing Switzerland, 2015, pp. 258–272. DOI: 10.1007/978-3-319-23267-6_17. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-319-23267-6_17/ (cit. on pp. 11, 57).
- [B8] —, "Energie Besparen in Datacentra", *Bits en Chips*, vol. 9, pp. 44–45, 2015. [Online]. Available: <http://bits-chips.nl/artikel/energie-besparen-in-datacentra/> (cit. on p. 11).
- [B9] B. R. Haverkort and B. F. Postema, "Towards Simple Models for Energy-Performance Trade-Offs in Data Centres", in *Proc. of Int. Workshop on Demand Modeling and Quantitative Analysis of Future Generation Energy Networks and Energy Efficient Systems*, 2014, pp. 113–122. [Online]. Available: <http://d-nb.info/1058949322/34/> (cit. on p. 39).
- [B10] B. F. Postema and B. R. Haverkort, "Stochastic Petri Net Models for the Analysis of Trade-Offs in Data Centres with Power Management", in *Proc. of the 3rd Int. Workshop on Energy-Efficient Data Centres (E2DC)*, S. Klingert, M. Chinnici and M. Rey Porto, Eds., ser. Lecture Notes in Computer Science, vol. 8945, Springer International Publishing, 2014, pp. 52–67. DOI: 10.1007/978-3-319-15786-3_4. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-319-15786-3_4/ (cit. on pp. 11, 37).

About the author

Björn Frits Postema was born on March 28, 1989 in Elburg, the Netherlands. From an early age, he was intrigued by programming and introducing his friends to the most innovative ways of using a computer. In the years after, he became a web developer. He went to Enschede in 2007, and received in 2011 his B.Sc. diploma for Technical Informatics at the UNIVERSITY OF TWENTE. He continued with a master in Computer Science in the track *Methods and Tools for Verification* (MTV) with the final topic *Fluid Survival Tool: A Model Checker for Hybrid*

Petri Nets at the UNIVERSITY OF TWENTE, and received his M.Sc. diploma in 2013. The MTV track was meant for ambitious computer science students, who aspired to become a validation engineer, a researcher and/or a tool builder. During his master, he obtained work experience during an internship with the company BETTERBE on the subject of modelling data centres.

In November 2013, Björn joined the *Design and Analysis of Communication Systems* (DACS) research group at the UNIVERSITY OF TWENTE and began his Ph.D. research on the topic of model-based analysis of energy-efficient data centres. The DACS research group has a multidisciplinary team with broad experience in methods and techniques to support design and dimensioning of dependable networked systems. His work was supported by the *Netherlands Organisation for Scientific Research* (NWO) project *Cooperative Networked Systems* (CNS), as part of the program *Robust Design of Cyber-Physical Systems* (CPS), which included the industrial partners TARGET HOLDING and BETTERBE.

For more information about me, please visit <http://bjornpostema.nl>.