Boston University

OpenBU **http://open.bu.edu**

Theses & Dissertations Boston University Theses & Dissertations

2016

Improving data center efficiency through smart grid integration and intelligent analytics

---

https://hdl.handle.net/2144/19505

Boston University

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

IMPROVING DATA CENTER EFFICIENCY THROUGH SMART GRID INTEGRATION AND INTELLIGENT ANALYTICS

by

HAO CHEN

B.S., Zhejiang University, 2010

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

2016

© 2016 by
HAO CHEN
All rights reserved

Approved by

First Reader

Ayse K. Coskun, Ph.D.
Associate Professor of Electrical and Computer Engineering

Second Reader

Michael C. Caramanis, Ph.D.
Professor of Mechanical Engineering
Professor of Systems Engineering

Third Reader

Ioannis Paschalidis, Ph.D.
Professor of Electrical and Computer Engineering
Professor of Systems Engineering
Professor of Biomedical Engineering

Fourth Reader

Sherief Reda, Ph.D.
Associate Professor of Engineering
Brown University

Logic will get you from A to B.
Imagination will take you everywhere. Albert Einstein

Acknowledgments

During my PhD study in Boston University, I am lucky to have met so many great people who have given me assistance, friendship, and encouragement along the way.

First and foremost, I would like to express my deepest gratitude to my PhD advisor, Prof. Ayse Coskun, for her inspirational guidance and constant support throughout my PhD study. I am always motivated by her enthusiasm in scientific research as well as in other aspects of life. I cannot really point out what I learned the most from her since it would be too much to say. She has been not only a supportive advisor but also a role model for me.

I am also very grateful to my PhD co-advisor, Prof. Michael Caramanis, for his guidance and valuable feedback on my research as well as his helpful career advice and suggestions. He has shown me the wonders of science, the fun and pleasure in working hard, and how to respect new ideas. I feel very fortunate and honored that I have worked with him.

I sincerely thank my dissertation committee members, Prof. Ioannis Paschalidis, Prof. Sherief Reda, and Prof. Orran Krieger for taking their precious time and giving invaluable comments on my dissertation. My dissertation could not be at this level without their advice.

I would like also to express my sincerest appreciation to Dr. Sastry Duri, Dr. Canturk Isci, and Dr. Vasanth Bala for their advice and support during my internships at IBM T.J. Watson Research Center. I would also like to thank Meng Wang, Yi Li, Dr. Yushan Chen, Tianqiang Liu, and Xing Meng for their encouragement, support, and company at Orbeus.

I am grateful to our collaborators and co-authors: Prof. Adam Wierman at Caltech, Prof. Zhenhua Liu at Stony Brook University, Dr. Bowen Zhang, Dr. Ata Turk, Dr. Can Hankendi and Ozan Tuncer at Boston University for their productive collaboration and all the stimulating discussions.

I was very fortunate to work with my friends, roommates, and colleagues in the PeacLab and also other labs around the world. Many thanks to all of them for all the inspiring discussions and great time we had together. I am also grateful to those who made my PhD study at BU a pleasant journey: Cali Stephens, Prof. Alan Pisano, Prof. Babak Kia, and many others.

Finally, I would like to thank my family. I owe my deepest gratitude to my parents for bringing me up, for understanding me, for providing the strongest and unconditional support, and for always giving me freedom to pursue what I want. I am also very grateful to my dear Jia He for her constant love, support and company. I would like to dedicate this dissertation to them.

The research that forms the basis of this dissertation has been partially supported by NSF grant 1038230, Sandia National Laboratories, VMware, Inc., Oracle Corporation, Decision Detective Corporation, and IBM Corporation.

The contents of Chapter 3 are in part reprints of the material from the papers, *Hao Chen, Ayse K. Coskun and Michael C. Caramanis, "Real-Time Power Control of Data Centers for Providing Regulation Service"*, in Proceedings of Conference on Decision and Control (CDC), 2013, *Hao Chen, Can Hankendi, Michael C. Caramanis and Ayse K. Coskun, "Dynamic Server Power Capping for Enabling Data Center Participation in Power Markets"*, in Proceedings of International Conference on Computer-aided Design (ICCAD), 2013, *Hao Chen, Michael C. Caramanis and Ayse K. Coskun, "The Data Center as a Grid Load Stabilizer"*, in Proceedings of Asia and South Pacific Design Automation Conference (ASPDAC), 2014, *Hao Chen, Michael C. Caramanis and Ayse K. Coskun, "Reducing the Data Center Electricity Costs Through Participation in Smart Grid Programs"*, in Proceedings of International Green Computing Conference (IGCC), 2014, *Hao Chen, Zhenhua Liu, Ayse K.*

Coskun and Adam Wierman, "Optimizing Energy Storage Participation in Emerging Power Markets", in Proceedings of International Green and Sustainable Computing Conference (IGSC), 2015, and *Hao Chen, Bowen Zhang, Michael C. Caramanis and Ayse K. Coskun, "Data Center Optimal Regulation Service Reserve Provision with Explicit Modeling of Quality of Service Dynamics"*, in Proceedings of Conference on Decision and Control (CDC), 2015.

The contents of Chapter 5 are in part reprints of the material from the papers, *Hao Chen, Sastry S. Duri, Vasanth Bala, Nilton T. Bila, Canturk Isci and Ayse K. Coskun, "Detecting and Identifying System Changes in the Cloud via Discovery by Example"*, in Proceedings of International Conference on Big Data, 2014, and *Hao Chen, Ata Turk, Sastry S. Duri, Canturk Isci, and Ayse K. Coskun, "Automated System Change Discovery and Management in the Cloud"*, in IBM Journal of Research and Development, 2015.

IMPROVING DATA CENTER EFFICIENCY THROUGH SMART GRID INTEGRATION AND INTELLIGENT ANALYTICS

HAO CHEN

Boston University, College of Engineering, 2016

Major Professors: Ayse K. Coskun, Ph.D.
Associate Professor of Electrical and Computer Engineering

Michael C. Caramanis, Ph.D.
Professor of Systems Engineering
Professor of Mechanical Engineering

ABSTRACT

The ever-increasing growth of the demand in IT computing, storage and large-scale cloud services leads to the proliferation of data centers that consist of (tens of) thousands of servers. As a result, data centers are now among the largest electricity consumers worldwide. Data center energy and resource efficiency has started to receive significant attention due to its economical, environmental, and performance impacts. In tandem, facing increasing challenges in stabilizing the power grids due to growing needs of intermittent renewable energy integration, power market operators have started to offer a number of demand response (DR) opportunities for energy consumers (such as data centers) to receive credits by modulating their power consumption dynamically following specific requirements.

This dissertation claims that data centers have strong capabilities to emerge as major enablers of substantial electricity integration from renewables. The participation of data centers into emerging DR, such as regulation service reserves (RSRs), enables the growth of the data center in a sustainable, environmentally neutral, or even beneficial way, while also significantly reducing data center electricity costs. In this dissertation, we first model data center participation in DR, and then propose runtime policies to dynamically modulate data center power in response to independent system operator (ISO) requests, leveraging advanced server power and workload management techniques. We also propose energy and reserve bidding strategies to minimize the data center energy cost. Our results demonstrate that a typical data center can achieve up to 44% monetary savings in its electricity cost with RSR provision, dramatically surpassing savings achieved by traditional energy management strategies. In addition, we investigate the capabilities and benefits of various types of energy storage devices (ESDs) in DR. Finally, we demonstrate RSR provision in practice on a real server.

In addition to its contributions on improving data center energy efficiency, this dissertation also proposes a novel method to address data center management efficiency. We propose an intelligent system analytics approach, "discovery by example", which leverages fingerprinting and machine learning methods to automatically discover software and system changes. Our approach eases runtime data center introspection and reduces the cost of system management.

Contents

1 Introduction **1**

1.1 Data Center Sustainability 2

1.2 Emerging Opportunities in Smart Grid 2

1.3 The Contributions and Significance of the Dissertation 3

1.4 Other Aspects of Data Center Management 7

1.5 Organization 9

2 Background and Related Work **10**

2.1 Power Markets and Capacity Reserves 10

2.2 Demand Response (DR) Programs 11

2.3 Power and Workload Management in Data Centers 17

2.3.1 Server Power Management 17

2.3.2 Data Center Level Power Management 18

2.3.3 Workload Scheduling and Control 19

2.4 Data Centers Participation in DR 21

2.5 Energy Storage Devices (ESDs) in DR 22

2.6 Distinguishing Aspects from Prior Work 23

3 Data Center Demand Response **25**

3.1 Overview 25

3.2 The Model of Data Center in DR 29

3.2.1 The Model of Servers 29

3.2.2 The Model of the Computational Unit 33

3.2.3 The Model of Data Center Workloads 35

3.3 An Initial Study: Single Server RSR Provision 37

3.3.1 Runtime Policy and Optimal Bidding 37

3.3.2 Experimental Results 39

3.4 The Best Tracking Runtime Policy 45

3.4.1 Policy Details 45

3.4.2 Energy and Reserve Bidding 48

3.4.3 Experimental Results 50

3.5 Comparison of Energy Cost Saving Strategies 54

3.6 The Stochastic Dynamic Programming (DP) Runtime Policy 61

3.6.1 The Formulation of Stochastic DP Problem 62

3.6.2 Policy Details 68

3.6.3 Energy and Reserve Bidding 71

3.6.4 Policy Comparison 72

3.7 The EnergyQARE Runtime Policy 73

3.7.1 Policy Details 74

3.7.2 Energy and Reserve Bidding 87

3.7.3 Experimental Results 89

3.8 A Prototype Implementation of Data Center RSR 100

3.8.1 Real Server RSR Provision Capability Test 101

3.8.2 Power - Resource Limits Model 102

3.8.3 Runtime Control Knobs and Policy Implementation 103

3.8.4 Real Server RSR Signal Tracking Performance 104

3.9 Energy Storage Devices (ESDs) in DR 106

3.9.1 Background on ESDs 106

3.9.2 Modeling ESDs 107

3.9.3 Market Opportunities for ESDs 110
3.9.4 Discussion 122
3.9.5 Managing Participation of ESDs in RSR 123
3.9.6 Comparison of Data Centers with ESDs in RSR Provision . . 129
3.10 Summary 130

4 Open Problems in Data Center DR 132

4.1 Optimization with Numerical and Analytical Methods 132
4.2 DR Participation by Data Centers with ESDs 134
4.3 Real-life Implementation of DR on Data Centers 135

5 Software and System Discovery in Data Center Cloud 138

5.1 Overview 138
5.2 Related Work 141
5.3 The Framework of Discovery 143
5.4 Change Set Creation 144
5.5 Fingerprint Creation and Learning 147
5.5.1 Fingerprint Creation 147
5.5.2 Learning with Fingerprints 152
5.6 Discovery by Examples 154
5.6.1 Experimental Methodology 155
5.6.2 Results 157
5.7 Summary and Open Problems 159

6 Conclusions 163

References 166

Curriculum Vitae 178

List of Tables

3.1 Descriptions of symbols on server power states. 33

3.2 The electricity monetary costs via different workload types and ($\bar{P}$, R), in single server RSR provision. 43

3.3 Performance comparison of optimal RSR provision and provision without regulation on single server. 44

3.4 Performance and savings of data center RSR provision with different types of workloads using the best tracking policy. 54

3.5 The hourly electricity bill of a 1000-server data center with RSR provision via different ($\bar{P}$, R), using the stochastic DP policy. 73

3.6 Descriptions of major symbols in Eq. (3.28) to Eq. (3.48). 77

3.7 Workload properties in experiments - Trace 1. 89

3.8 Workload properties in experiments - Trace 2. 89

3.9 Workload properties in experiments - Trace 3. 90

3.10 A selection of today's typical capacities of ESDs, based on space constraints. 115

3.11 Optimal solutions for ESDs in peak shaving. 120

3.12 Comparing the optimal net profit of multiple types of ESDs (with E_{cap}, P_{cap} listed in Table 3.10) in participating different DR programs. . . 123

3.13 Capacities and costs of ESDs in provision of 550 kW RSR. 130

List of Figures

2·1 Typical PJM 150sec ramp rate (F) and 300sec ramp rate (S) RSR signal trajectories (PJM, 2016). 15

3·1 Data center DR participation framework. 26

3·2 The model of data center in DR participation. 30

3·3 The relation between server power and throughput (in RIPS) for applications in PARSEC-2.1 benchmark suite. Dots are the real measurements and lines are the linear fitting curves. 32

3·4 Results of Blackscholes with $\bar{P} = 117.65W$ and $R = 30W$ in single server RSR provision. (i) $P_{tgt}(t) = \bar{P} + y(t)R$ and $P_{con}(t)$ trajectories (in Watts) over a 11-hour period (10 replications of a 1-hour period, the first hour data is not used because of the warming up process). The tracking error statistics: $\bar{\epsilon} = 0.20$, $\sigma_\epsilon = 0.60$. (ii) HPQ QoS degradation for each job arrival shown as a red dot on the time trajectory. The overall statistics: $\bar{D}^H_{bls} = 2.91$, $\sigma_{D^H_{bls}} = 1.22$. (iii) LPQ QoS degradation for each job arrival shown as a red dot on the time trajectory. The overall statistics: $\bar{D}^L_{bls} = 11.27$, $\sigma_{D^L_{bls}} = 9.31$. 41

3·5 Results of mixed workload of Blackscholes and Canneal in single server RSR provision with $\bar{P} = 115.33W$ and $R = 30W$. (i) $P_{tgt}(t) = \bar{P} + y(t)R$ and $P_{con}(t)$ trajectories (in Watts). The tracking error statistics: $\bar{\epsilon} = 0.20$, $\sigma_{\epsilon} = 0.58$. (ii) HPQ QoS degradation trajectory with overall statistics: for Blackscholes $\bar{D}^{H}_{bls} = 3.53$, $\sigma_{D^{H}_{bls}} = 2.02$ and for Canneal $\bar{D}^{H}_{can} = 2.64$, $\sigma_{D^{H}_{can}} = 1.01$. (iii) LPQ QoS degradation trajectory with overall statistics: for Blackscholes $\bar{D}^{L}_{bls} = 16.84$, $\sigma_{D^{L}_{bls}} = 11.63$ and for Canneal $\bar{D}^{L}_{can} = 8.36$, $\sigma_{D^{L}_{can}} = 5.77$. 42

3·6 The flowchart of the best tracking policy. 46

3·7 The probability density function (PDF) of tracking error (a, b, c) and job QoS degradation (d, e, f) in different cases with the best tracking policy. All the cases are with the (homogeneous) workload set of Blackscholes. 50

3·8 Comparison of hourly energy consumption, peak power and monthly electricity bill of different energy cost reduction strategies in various scenarios. (a), (b), and (c) are results under different data center utilization U = 20%, 50% and 80%; (d), (e), and (f) are results with different workload types, i.e., Blackscholes and Streamcluster; (g), (h), and (i) are results with servers using shallow or deep sleep states. . . 59

3·9 Mean of the QoS degradation is characterized by the integration of the past power consumption. Strong anti-correlation -0.97 is found between two curves. 64

3·10 Variance of QoS degradation is characterized by the number of job departure (i.e., finished) in 4 seconds. The higher departure number results in higher uncertainty to the system, and thus higher variance. 66

3·11 The optimal stochastic DP policies $u(t)$ via $y(t)$ and $z(t)$ given $d(t) = 1$, of three cases: (a) $PC_{track}(t) >> PC_{QoSD}(t)$, i.e., the tracking cost dominates in the overall period cost function; (b) $PC_{track}(t) << PC_{QoSD}(t)$, i.e., the cost of QoS degradation dominates in the overall period cost function; (c) $PC_{track}(t)$ and $PC_{QoSD}(t)$ are on the same order of magnitude. 69

3·12 The flowchart of the EnergyQARE runtime policy. 75

3·13 Mean of the signal tracking error and the QoS degradation via different $\bar{P}$ and R (normalized to the maximal possible value R_{max} given $\bar{P}$) in EnergyQARE. (a) and (b) are results of the workload trace 1; (c) and (d) are results of the workload trace 2. (b) is for the Canneal application in trace 1 and (d) is for the Dedup application in trace 2. 91

3·14 Results of RSR signal tracking and QoS degradation for the workload trace 1 with $(\bar{P}, R)$ at their optimal values in EnergyQARE. (a) is the real dynamic power $P_{con}(t)$ compared with the signal power cap $P_{tgt}(t)$ during the 1-hour simulation. (b) is the cumulative distribution function (CDF) of the tracking error $\epsilon(t)$. The green lines show the tracking error probabilistic constraints, i.e., $(\epsilon^{tol}, \eta^{\epsilon})$. (c) and (d) are the CDF of the QoS degradation of Canneal and Freqmine applications, respectively. The green lines are the SLAs, i.e., (Q_j, η_j). 93

3·15 The impact of the size of the feedback window T to the tracking error and workload QoS in EnergyQARE. The probabilities of the tracking error and the QoS degradation that are smaller than ϵ^{tol}, and Q_j for Canneal and Freqmine applications respectively, are shown in the figure. 94

3·16 The energy monetary costs in three scenarios: “optimal RSR”, ”fixed cap” and “without cap” in different cases. All the costs are normalized to the largest value in each figure. In (b) we also calculate the absolute value of the cost *savings* from the “optimal RSR” scenarios to their corresponding “fixed cap” scenarios, and represent the absolute savings in the black line. 96

3·17 The CDF of tracking error and QoS degradation of both the EnergyQARE and the best tracking policy for workload trace 1. The blue curves represent the best tracking policy, and the red curves represent the EnergyQARE. The green lines represent the tracking error and the workload SLA probabilistic constraints, i.e., $(\epsilon^{tol}, \eta^{\epsilon})$ and (Q_j, η_j). . . . 97

3·18 Power- CPU resource limits models for the PARSEC applications. . 101

3·19 The overview of the runtime power capping technique. Our technique receives input from the ISO and the VM (e.g., CPU_{used}, etc.) to make CPU resource limit adjustments so as to keep the power consumption close to the current power cap. 103

3·20 1-hour RSR signal power capping of Streamcluster application by adjusting the CPU resource limits. We both show the real power consumption and the power cap values (top figure) and the dynamic adjustment of CPU resources (bottom figure). 104

3·21 1-hour power profile of the server running the heterogeneous workloads when we apply our proposed power capping technique. 105

3·22 ESDs in RSR provision. (a) and (b) show the optimal net profit via varying energy and power capacities for LI batteries and UCs; (c) and (d) show the optimal net profit via varying amount of reserve provision, and the varying reserve price Π^R, respectively for various ESDs. The black dashed line in (d) shows the current Π^R. 114

3·23 The optimal net profit via varying contingency reserve prices Π^{CR} for ESDs with today's typical capacities. The black dashed line shows the current Π^{CR}. 117

3·24 ESDs in peak shaving. (a) is an example of the daily power curve before peak shaving; (b) (c) and (d) are the optimal net profit via varying energy and power capacities (E_{cap}, P_{cap}), for LI batteries, UCs and CAES respectively; (e) and (f) are the optimal net profit via varying cap-ex and op-ex peak power prices respectively for multiple ESDs. The black dash lines show where the current market prices are around. 121

3·25 The revenue of providing RSR via varying ρ_2, for LI batteries (in (a)) and UCs (in (b)), with three heuristic offline solutions, respectively. The revenue is normalized to the value of $\rho_2 = 1$. 125

3·26 The revenue of providing RSR via varying ρ_2, for LI batteries (in (a)) and UCs (in (b)), respectively, with offline and online solutions. The revenue is normalized to the value of $\rho_2 = 1$ in the offline solution. . 129

4·1 The communication between the master and slave nodes. 136

5·1 Flowchart of change set creation. Snapshots of the system are captured before and after the system change event. Then, a diff operation is calculated on these two snapshots, and the change set is generated. . 144

5·2 Training and discovery phases of the system change discovery framework. Labels and extracted fingerprints from change sets are input to learning algorithms to train the model in the training phase. The learned model is then used to discover and label the new-coming unidentified changes during discovery. 145

5·3 A sample change set. It contains all features that are created, modified or deleted during the system change event, e.g., OS, files, packages, processes and configurations. 146

5·4 The flowchart of the histogram fingerprint generation. 150

5·5 Two-dimensional (2D) vectors created by w2v for a set of filenames when file tree information is supplied to it. Created vectors retain the semantic relationship among the software objects they represent. Vector dimensions are indicated by x and y. 151

5·6 Discovery accuracy for multiple fingerprinting methodologies and learning algorithms. Results are grouped by learning algorithms. 157

List of Abbreviations

ACE	......	Area Control Error
AWS	......	Amazon Web Service
CAES	......	Compressed Air Energy Storage
CDF	......	Cumulative Distribution Function
DP	......	Dynamic Programming
DR	......	Demand Response
DVFS	......	Dynamic Voltage and Frequency Scaling
ESD	......	Energy Storage Device
FC	......	Frequency Control
FCFS	......	First Come First Serve
FW	......	Flywheel
HPC	......	High Performance Computing
HPQ	......	High Priority Queue
ISO	......	Independent System Operator
LA	......	Lead-acid
LI	......	Lithium-ion
LPQ	......	Low Priority Queue
MOC	......	Massachusetts Open Cloud
MPC	......	Model Predictive Control
PDF	......	Probability Density Function
QoS	......	Quality of Service
RBF	......	Radial Basis Function
RIPS	......	Retired Instructions Per Second
RSR	......	Regulation Service Reserve
SLA	......	Service Level Agreement
SVM	......	Support Vector Machine
UC	......	Ultra/super-capacitor
UPS	......	Uninterruptible Power Supply
VM	......	Virtual Machine
Word2vec	......	w2v

Chapter 1

Introduction

Data centers are among the key enabling technologies for the rapidly evolving IT industry. The ever-increasing growth in the demand of IT computing, storage and large-scale cloud services leads to the proliferation of data centers that consist of hundreds to millions of servers. According to a 2011 report, there are more than 500,000 data centers worldwide (Miller, 2011). The number is predicted to continue growing to 8.6 million in 2017 (Sverdlik, 2014). The number of servers in large data centers has reportedly passed the 100,000 mark (Katz, 2009). Data centers have accounted for a global market size of 152 billion US dollars by 2016 (Dayarathna et al., 2016), and have become one of the largest worldwide electricity consumers. As a result, data center energy and resource efficiency has started to receive significant attention due to its economical, environmental and performance impacts.

This dissertation aims to improve data center energy and resource efficiency through enabling data centers to participate in smart grid *demand response* (DR) programs. The dissertation claims that data centers have strong capabilities to emerge as major enablers of substantial electricity integration from renewables into the grid. The participation of data centers into emerging DR, such as *regulation service reserves* (RSRs), enables the growth of the data centers in a sustainable, environmentally neutral, or even beneficial way, while also significantly reducing data center electricity costs.

1.1 Data Center Sustainability

Electricity used by data centers in the US accounts for around 3% of the total electricity consumption (Koomey, 2011), with an estimated growth rate of 12% per year (Rao et al., 2012). To put this in context, 3% of the US electricity production is about 120 billion kWh or equivalent to the average consumption of a large city with 11.6 million households. The steep increase in usage, along with the growth of the electricity price, double the electricity bill of a typical data center every five years (Dayarathna et al., 2016). Energy costs have become a significant portion of the overall data center cost of ownership today, and have even exceeded the hardware purchasing costs in some cases (Rivoire et al., 2007). Furthermore, the fast growth of data center energy usage has tremendous environmental impacts. A 2009 McKinsey Corporation report states that the world's 44 million servers produce 0.2 percent of all carbon dioxide emissions, or 80 megatons a year, approaching the emissions of entire countries like Argentina or the Netherlands (Katz, 2009), and the number has been growing fast. There is an urgent need to make the growth of data centers sustainable.

1.2 Emerging Opportunities in Smart Grid

Energy efficiency and environmental sustainability objectives of the whole society are pushing the integration of an aggressively growing amount of renewable energy generation (e.g., hydropower, wind power, and solar energy). Currently, the vast majority of electricity production comes from fossil fuels, which is long-term unsustainable and has a tremendous environmental impact. The EU has set the goal of reaching a 20% share of renewable energy in gross energy consumption by 2020 (Bohringer et al., 2009). In the US, 38 states have long term renewable portfolio standards and 14 states have installed more than 1,000 MW of wind power (AWEA, 2015). It is expected that the total renewable generating capacity will have a growth of 52% till 2040 in the US (EIA, 2014).

The volatility and intermittency of renewable generation, however, combined with the lack of reliable large-scale energy storage solutions, create challenges for grid independent system operators (ISOs) who need to match supply and demand by securing commensurate flexible capacity reserves in forward markets and dispatching them in real time. In response to this challenge, emerging ancillary power markets (e.g., PJM (PJM, 2016), NYISO (NYISO, 2016)) provide sizable monetary incentives for the consumers to perform DR, which refers to a consumer adjusting its own electricity usage following a set of constraints or directives given by ISOs.

Recent advanced server power management techniques, such as dynamic voltage and frequency control (DVFS) (Li and Martinez, 2006), power budgeting (Zhan and Reda, 2013) and workload management (Ghatikar, 2014) have enabled data centers to use the flexibility in their power consumption to manage cost and energy use. We envision that data centers offer a unique opportunity to participate in emerging DR programs, and it would be highly appealing if they were enabled to participate in these opportunities in practiced scenarios as well. By doing so, data centers can decrease a large portion of their energy costs, while helping satisfy most of the growth in data center energy consumption from the renewable energy, and also provide additional reserves to other less flexible uses of electricity in the society.

1.3 The Contributions and Significance of the Dissertation

Differing from a considerable body of prior research that has focused on reducing data center energy consumption and improving energy efficiency through intelligent power management techniques such as DVFS (Li and Martinez, 2006), workload consolidation (Teodorescu and Torrellas, 2008), power budgeting (Rajamani et al., 2006; Zhan and Reda, 2013), job scheduling (Mu'alem and Feitelson, 2001), or efficient data center cooling (Patel et al., 2003), our work mainly studies the integration of data center with smart grid through DR participation. While some prior work has studied the participation of data centers in legacy DR programs, such as dynamic energy pricing (Liu et al., 2014), peak shaving (Wang et al., 2012; Aksanli et al., 2013), and emergency demand reduction (Zhang et al., 2015; Tran et al., 2016; Islam et al., 2016), the demand side RSR provision is entirely new to data centers. The RSR market is especially of our interest because considerable monetary savings are easily anticipated for data centers as participants due to the high reserve market clearing prices, which are, on average, as valuable as energy clearing prices in today's markets (PJM, 2013; NYISO, 2016). We foresee that the monetary savings from the RSR markets could be several times higher than those from legacy DR programs. Furthermore, we focus on RSR provision because on one hand their requirements are expected to increase rapidly with increasing renewable energy integration in the grid (Makarov et al., 2009), while on the other hand data centers have a comparative advantage in offering RSRs relative to other demand side reserve providers.

Our work is the first to thoroughly study and evaluate the data center participation in RSR market, on both the capabilities and the profits, from multiple perspectives including software, hardware, math and control. We claim that data centers offer a unique opportunity to provide RSRs, which not only enables the growth of the data center in a sustainable, environmentally neutral, or even beneficial way, but also reduces data center energy monetary costs tremendously. Specifically, the contributions are as follows:

- We introduce *practical* models of the data center in RSR provision, which consider heterogeneities in workload (i.e., different types of application running), multiple server power states, the associated time delay and energy loss during server state transition, power budgeting, workload allocation and queuing, and the workload service level agreements (SLAs), etc. (Chen et al., 2013a; Chen et al., 2016a);

- We propose real-time dynamic policies targeting different scenarios in power management and workload servicing to modulate the data center power following the RSR signal requirement broadcast by ISOs. These policies are:
    1. The *best tracking* policy that tracks the RSR signal as accurately as possible. The policy is the first to enable data center level RSR provision on the practical data center model, and is suitable for the scenario with a tight signal tracking constraint but loose workload QoS constraints (Chen et al., 2014a);
    2. The *stochastic dynamic programing* (DP) policy that leverages the statistics of the RSR signal and the workload servicing performance in optimizing the tradeoff between the signal tracking and workload QoS. The policy is optimal and applicable on a simplified data center model (Chen et al., 2015c);
    3. The *EnergyQARE*, i.e., the energy and QoS-aware RSR enabler policy that builds upon a real-life practical data center model, which considers heterogeneities in workload, various server power states, the time delay and energy loss during server state transition, as well as workload SLAs. The policy not only enables data centers to track the RSR signal accurately, but also guarantees workload QoS constraints that are determined by SLAs. The policy is suitable for general and practical data center scenarios with tight workload QoS constraints (Chen et al., 2016a).
- We formulate an optimization problem to solve the optimal energy and reserve

bidding strategy in data center RSR provision to minimize the data center energy monetary cost, with constraints on RSR signal tracking requirements, workload SLAs, and system specifics (Chen et al., 2013a; Chen et al., 2016a);

- We evaluate the overall capabilities and profits of data center RSR provision, and make comparisons to other energy cost saving strategies. Our results demonstrate that data centers in a general scenario can achieve up to 44% energy monetary savings by providing RSRs compared to a regular energy use without any reserve provision, which are much higher than the savings from transitional energy cost reduction strategies (Chen et al., 2014b; Chen et al., 2016a);

- We implement the designed optimization framework and runtime policies of RSR provision on a real server as a prototype of the data center level implementation. This initial implementation provides guidance for the future deployment of our techniques onto real-life data centers for practical industrial uses (Chen et al., 2013b; Turk et al., 2016b);

- In addition to data centers, *energy storage devices* (ESDs) are also potential candidates for DR provision. In fact, some studies model data centers as large-scale ESDs, and evaluate the equivalent capacities of ESDs that data centers can offer (Liu et al., 2014). To better understand and compare data centers and ESDs in DR programs, in this dissertation we also investigate the capabilities and profits of different types of ESDs in participating various DR programs. This investigation also provides clues for future studies on the DR participation by the combination of data centers and their associated ESDs together (Chen et al., 2015a; Chen et al., 2015b).

1.4 Other Aspects of Data Center Management

In tandem with the growing challenges of data center energy and environmental sustainability, the expansion of the data center size leads to an increasingly complex management problem, including a great variety of issues from different sources, such as hardware failures, software vulnerabilities, network congestion, and malicious attacks. As a result, today's data centers are experiencing tremendous operation and management costs. According to the IDC study in 2012, approximately 70% of data center spending is on management and administration (Villars et al., 2012). In order to reduce the overall data center costs, emerging data center platforms require more efficient, scalable, automated and intelligent data center management and analytics solutions.

As part of this dissertation, we specifically target the problem of efficient software and system discovery, which plays a significant role in system management and analytics, problem detection, and diagnosis. A typical data center today hosts hundreds of thousands of instances (i.e., virtual machines (VMs) or containers). These instances evolve differently from the time they are booted. Consider the following scenario: we discover a vulnerability on one instance after a system update or a software installation, and we would like to understand how many other instances in the data center have similar update, as these instances may encounter the similar vulnerability. Moreover, the discovery needs to be fast and efficient, so that users can be early warned and problems can be solved in time. Traditionally, the system discovery is conducted with designed *rules*, which check for the existence of certain files and their attributes (OpenLogic, nd; OpenIOC, nd). The rule-based approaches, however, are fragile, require high expertise and constant maintenance, indicating a substantial amount of manual effort. A great amount of today's software is released or updated multiple times a week, and many systems change every day. Rule-based approaches have difficulties in keeping up with the pace of software and system changes. As a result, more scalable, automated and intelligent discovery approaches are essential in today's data center environment.

In this dissertation, we propose a novel direction: using *discovery by example* as an alternative solution to the rule-based approaches in software and system discovery. We believe the solution is substantially more efficient in data center and cloud management, as it is more generalized and scalable, and it is able to learn automatically and incrementally. The specific contributions are as follows:

- We introduce an automated system discovery and analytics solution for the cloud, "discovery by example", that generates fingerprints of changes in system state, and utilizes these fingerprints in a machine learning platform for software, system change discovery and management (Chen et al., 2014c);
- We propose multiple novel feature extraction methods, such as histogram and natural language processing based vectors (Mikolov et al., 2013a), to generate condensed fingerprints from the comprehensive metadata associated with software and system changes. The designed feature extraction methodologies primary focus on the file system features. They can learn the hidden context behind filenames, and represent them with vectors utilizing the file tree structure and/or file co-location information to capture the semantic relationships of files (Chen et al., 2016b);
- We build an adaptive knowledge-base that enables fast comparison of software and system changes with previously labeled data. Specifically, we learn the discovery models from the knowledge-base with learning algorithms and then predict new software and system changes using these models (Chen et al., 2014c; Chen et al., 2016b);

- We evaluate and compare the discovery speed and accuracy on a variety of feature extraction and machine learning methods. Our results show that our mechanism can be utilized for fast (in a few milliseconds or seconds) and accurate (up to 98.75%) discovery (Chen et al., 2016b).

1.5 Organization

The rest of this dissertation starts with a review of the background and related work on data center power and workload management, DR programs and ESDs in Chapter 2. Chapter 3 studies the data center participation in DR programs, especially the RSR. The data center participation model is first introduced, followed by our dynamic control policies and our optimal energy and reserve bidding strategy. The chapter then evaluates the performance and data center energy cost savings from RSR provision, and also compares the savings to those from other energy cost reduction strategies. After that, an implementation of the RSR provision framework and the runtime policies on a real system – a real server is introduced. At the end of Chapter 3, the capabilities and profits of different types of ESDs in various DR programs are evaluated. Chapter 4 discusses the open problems and future research directions in data center DR participation.

We believe a significant orthogonal problem to data center energy efficiency is the efficiency of software and system management in data centers. To the end, Chapter 5 investigates an intelligent analytics solution in data center management, i.e., the *discovery by example* approach, and its framework as an automated and scalable solution for software and system vulnerability discovery. Chapter 6 summarizes the dissertation.

Chapter 2

Background and Related Work

2.1 Power Markets and Capacity Reserves

Power markets, introduced in the US in 1997 (Ott, 2003), have been widely adopted. Today they serve the majority of high-voltage-connected generators and large consumers. Soon after their introduction, power markets evolved to co-optimize or co-clear energy and capacity reserves, whose system-level requirements reflect contingency planning for uncertainty in energy balance, transmission, and generating capacity availability. Social-welfare contributions of competitive power markets are arguably due to the fact that they enable distributed, yet collaborative, decisions which (i) take advantage of locally known uncertainty and dynamical-response-capability information, and (ii) can respond efficiently to price or other system-wide state sufficient statistics, such as frequency and Area Control Error (ACE) and associated reserve requirement signals. These sufficient statistics enable local decisions to be made efficiently and in a manner that is adaptive to power system requirements.

Synchronized power systems may become unstable when generation and consumption are not carefully balanced in practically real-time. To this end, Independent System Operators (ISOs) solicit and secure sufficient quantities of a mix of reserves with different dynamic delivery properties. Bi-directional reserve contracts are secured at least an hour in advance and promise to respond in real-time to ISO-broadcasted fast changing system requirements.

Each type of reserves is characterized by the time scale and the frequency of the reserve commands deployed. For the time-scale, focusing on the short-term markets that are most relevant to this work, there are (i) day-ahead markets that close at noon of the previous day and clear energy and reserve bids for each of the 24 hours of the next day, (ii) hour-ahead adjustment markets that close an hour in advance of each hour, allowing participants to adjust their day ahead positions on both energy and reserves at clearing prices that reflect the new information, and (iii) 5-minute close-to-real-time economic dispatch markets that determine ex post marginal cost of energy employed to adjust participant revenues and costs for deviating from the quantities cleared in the previous two markets (Kranz et al., 2003; NYISO, 2016; Ott, 2003). Based on different frequencies of the reserve commands deployment, there are primary (or frequency control, i.e., FC), secondary (i.e., regulation service) and tertiary reserves, in which reserve requests are deployed respectively in millisecond, second and minute intervals (PJM, 2016).

Capacity reserves have been offered primarily by centralized generators, but market rules are changing to allow the demand side to offer reserves as well. For example, PJM, one of the largest US ISOs, has allowed electricity loads to participate in reserve transactions since 2006 (PJM, 2005), with other ISOs contemplating to follow the suit. Demand side capacity reserves, as an emerging type of demand response (DR), is starting to play a significant role in stabilizing power systems, and is particularly beneficial as intermittent and volatile renewable generation is integrated at ever increasing rates. Next, we review both legacy and emerging DR opportunities.

2.2 Demand Response (DR) Programs

Demand response (DR) refers to electricity consumers regulating their power usage following market requirements. Widely studied DR programs pertain to a few legacy programs such as dynamic energy pricing (Wierman et al., 2014; Zhu et al., 2013), peak shaving (Govindan et al., 2011; Wang et al., 2012), and emergency load reduction (Zhang et al., 2015; Tran et al., 2016; Islam et al., 2016). In dynamic energy pricing, the demand side modulates its power consumption so as to consume more power at the valley of the energy price and less as prices peak. Medium to large commercial and industrial power consumers are often under *coincident peak pricing* rates that charge a very high cost for usage during the hour that is coincident to the system peak hour (FortCollins, nd). In addition to charges on energy, these medium to large power consumers are also charged for their peak power over an agreed upon period, e.g., over a month (Govindan et al., 2011). Sometimes there are even strict limits on the peak power consumed during periods with shortage of supply. In these cases, limiting the peak power, known as peak shaving (Wang et al., 2012), has been used to reduce costs and enable stability of power systems. In emergency load reduction, the power market operator coordinates large electricity consumers for load reduction in emergency situations, in order to prevent major economic losses and catastrophic events such as blackouts (Zhang et al., 2015).

Recently, power markets start to allow demand side to provide capacity reserves as an emerging DR. In demand side capacity reserves, power consumers complement generators in buying of energy and offering all kinds of capacity reserves in dynamic market. Thus, consumers are obliged to regulate their power consumption to track some dynamic power targets based on the amount of reserve that they have offered in the market time scale (Hansen et al., 2014). As introduced before, there are mainly three types of reserves. We consider all three types of power capacity reserves under the following notation: primary reserves or FC, R_1, secondary reserves or RSR, R_2 and tertiary or contingency reserves, R_3. Providers are obliged to modulate their power consumption so as to track a stochastic non-anticipatory dynamic power target, $P_{tgt,i}(t)$ for $i \in 1, 2, 3$. For primary and secondary reserves, the target varies symmetrically about a fixed average power level $\bar{P}_i$ allowing energy neutral time averaged consumption. Although $P_{tgt,i}(t)$ dynamics are stochastic and are revealed to reserve providers only with short notice, their statistical behavior is well known.

Primary Reserves or FC

A primary reserve provider that has offered R_1 in the hour ahead market, must modulate its power consumption $P_{con}(t)$ in real time to track a target $P_{tgt,1}(t)$ that is determined as a function of the local (and hence fully distributed) frequency measurement $\omega(t)$. Denoting frequency deviations from 60Hz by $\Delta\omega(t) = \omega(t) - 60$, we have $P_{tgt,1}(t)$ as follows:

$$P_{tgt,1}(t) = \begin{cases} \bar{P}_1 - R_1, & \Delta\omega(t) \leq -0.2, \\ \bar{P}_1 + \frac{(\Delta\omega(t)+0.02)}{0.2-0.02}R_1, & -0.2 < \Delta\omega(t) < -0.02, \\ \bar{P}_1, & |\Delta\omega(t)| \leq 0.02, \\ \bar{P}_1 + \frac{(\Delta\omega(t)-0.02)}{0.2-0.02}R_1, & 0.02 < \Delta\omega(t) < 0.2, \\ \bar{P}_1 + R_1 & \Delta\omega(t) \geq 0.2. \end{cases}$$

$P_{tgt,1}(t)$ is a piecewise linear function of $\Delta\omega(t)$, representing the local impact of system-wide supply-demand imbalances. Under most circumstances the statistical behavior of $\Delta\omega(t)$ constitutes a zero mean white noise, whose variance is well known at the beginning of the hour.

In FC, $P_{tgt,1}(t)$ varies in real time according to $\Delta\dot{\omega}(t)$. We approximate the real time dynamics of $\omega(t)$ by discrete time dynamics with a small time increment of 0.1 seconds. $\Delta\dot{\omega}(t)$ is generally unconstrained, but $\dot{P}_{con}(t)$ is of constant magnitude. More precisely, $\dot{P}_{con}(t) = SGN(P_{tgt,1}(t) - P_{con}(t))R_1/30$ MW/sec. When $P_{con}(t) = P_{tgt,1}(t)$ then and only then $\dot{P}_{con}(t) = 0$. As such, there is no tracking error allowed in FC reserve power output modulation, as is instead the case with secondary reserves. Although primary reserves are not yet cleared in power markets, and are in fact provided by centralized generation facilities (Zhao et al., 2012), in anticipation of markets evolving in this direction, we assume for purposes of obtaining a reasonable estimate of primary reserve clearing prices, Π^{R_1}, that Π^{R_1} equals several times of the value of energy clearing prices Π^E. This is a reasonable assumption given that primary reserves are more valuable than secondary reserves where we assume that Π^{R_2} is of the same order of magnitude as Π^E. In the numerical results reported in the later chapters, we use the relationship $\Pi^E \bar{P}_1 - \Pi^{R_1} R_1$ to evaluate the effective energy cost of a data center that offers FC reserves R_1.

Secondary Reserves or RSR

A significant difference compared to primary reserves of RSR is that each provider is obligated to track the same relative target determined generally by an ISO signal that we denote by $y(t)$. In fact, $y(t)$ is the output of an ISO specified integral proportional filter of the ACE (that measures the difference between actual and scheduled net imports from adjacent balancing areas) and frequency excursions outside of the tolerance interval (i.e., $[59.980, 60.020Hz]$). It is unpredictable and unaffected by behavior of any individual market participant. The statistical behavior of $y(t)$, however, is known ahead. It is a zero mean scalar taking values in the interval $[-1, 1]$, and follows a well behaved two level Markov model whose transition probabilities can be usually calibrated a few hours in advance. The signal is centrally determined and broadcasted every 4 seconds by the ISO, with the increments in each 4 seconds not exceeding $\pm R_2/(\tau/4)$ where τ is 150 seconds for the fast (F) RSR and 300 seconds for the slower (S) RSR (PJM, 2016). Figure 2·1 depicts actual historical data trajectories of $y(t)$ corresponding to two different normalized speeds of 1/150 and 1/300 MW/sec.

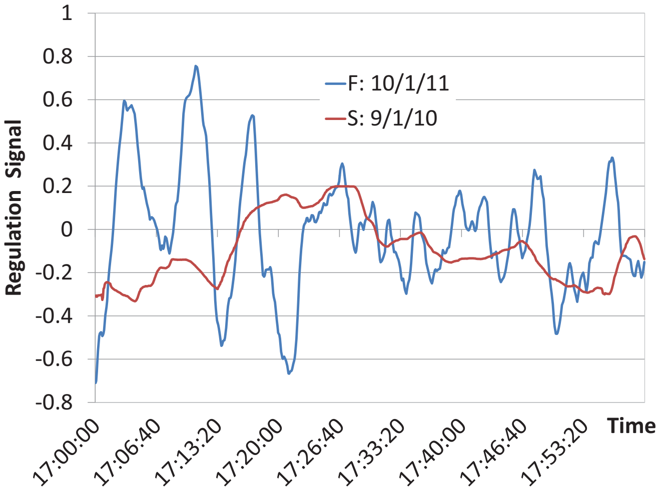


Figure 2·1: Typical PJM 150sec ramp rate (F) and 300sec ramp rate (S) RSR signal trajectories (PJM, 2016).

An RSR provider who has offered R_2 in the hour ahead market is obliged to modulate its power consumption $P_{con}(t)$ to track the target $P_{tgt,2}(t) = \bar{P}_2 + y(t)R_2$ at a constant, albeit slower, speed than FC. With energy and reserve market clearing prices, Π^E and Π^{R_2}, that are of similar value, an RSR provider sees an effective energy cost of $\Pi^E \bar{P}_2 - \Pi^{R_2} R_2$. The credit received may be further reduced as a function of the tracking error $\epsilon(t)$ (usually of its statistics, such as the mean $\bar{\epsilon}$ of it), which is measured during the hour as[1]:

$$\epsilon(t) = \frac{|P_{con}(t) - P_{tgt,2}(t)|}{R_2}. \tag{2.1}$$

Furthermore, the reserve provider may lose its contract in further RSR provision, if the tracking error $\epsilon(t)$ exceeds a probabilistic tolerance constraint $(\epsilon^{tol},\ \eta^{\epsilon})$, i.e.,:

$$Probability\ \{\epsilon(t) > \epsilon^{tol}\} > 1 - \eta^{\epsilon}. \tag{2.2}$$

---

[1] Since this dissertation mainly studies RSR provision, for simplicity, by default we use R, $P_{tgt}(t)$, $\bar{P}$, and Π^R to denote the parameters in RSR, i.e., R_2, $P_{tgt,2}(t)$, $\bar{P}_2$, and Π^{R_2} in the following chapters.

Tertiary or Contingency Reserves

Tertiary reserve provision is typically scheduled in the 5-minute power markets. It involves rescheduling of the provider's consumption from a pre-contingency or pre-congestion level $\bar{P}_3$ to a post contingency or post congestion level that is as much as R_3 lower. That lower level must then be maintained for up to a few hours. The speed at which tertiary reserves must be offered is far slower than that of primary or secondary reserves. Tertiary reserves are often operated with *load migration* (Wang et al., 2014; Chiu et al., 2012; Liu et al., 2011) discussed further below.

Significance of Data Centers in Emerging DR

It is the aforementioned role of providing emerging DR, i.e., the capacity reserves we envision for data centers, and more broadly, for computing systems. In today's grid, the FC and RSR needed to ensure stability by guaranteeing tolerable ACE and frequency deviation errors amount to about 0.1% for FC and 1% for RSR related to the total electricity load. This amount is in fact comparable to the 2-3% figure attributed to electricity consumption by data centers (Koomey, 2008).

Considering today's market conditions, secondary reserves are traded at a price comparable to the price of energy, while, if primary reserves are introduced into power markets, they will probably command higher clearing prices. This implies that a data center able to provide RSRs equal to 50%, or alternatively FC reserves equal to 10%, of its average energy consumption, may be able to reduce its energy cost by up to 50%. There are, therefore, substantial increasing economic incentives for data center operators to participate in jointly clearing energy and reserve markets. In addition, the associated societal and sustainability benefits that may result from greater adoption of renewables enabled by effective data center reserve provision are also clearly enormous.

In the following sections, we overview the state-of-the-art power and workload management techniques of data centers as well as the data center participation in DR programs.

2.3 Power and Workload Management in Data Centers

Power management techniques in both server level and data center level, along with the workload management in the data center have all been advanced significantly in recently years. These techniques provide the data centers with the capability to modulate their power at fine granularity. In this section, we survey the state-of-the-art techniques in these areas.

2.3.1 Server Power Management

The majority of the processors today are designed to support various energy-aware operation settings (Burd and Brodersen, 1995). Widely used control knobs include dynamic voltage-frequency scaling (DVFS) and power gating features to turn off idle units (Li and Martinez, 2006). Multi-core processors offer additional degrees of freedom for managing power through workload allocation (Teodorescu and Torrellas, 2008). Recently, voltage and frequency islands have been introduced for achieving fine-grained system level power management (Ogras et al., 2007).

Dynamic power management at the processor level typically focuses on designing efficient techniques to put idle units into sleep states while minimizing the performance overhead from switching between states (Benini et al., 2000). PowerNap is a similar approach at the server level for eliminating the server idle power and reducing the state transition overhead (Meisner et al., 2009). Isci et al. (Isci et al., 2013) explore the feasibility of low-latency power states implemented at the server hardware and introduce a power-aware virtualization management policy.

Today's systems also employ power capping mechanisms to prevent the power from exceeding the peak power constraints. DVFS is a popular control knob for capping (Fan et al., 2007). For multi-threaded applications, DVFS can be combined with thread allocation and migration to perform finer granularity power capping (Cochran et al., 2011; Rangan et al., 2009).

As the virtualization technique has advanced significantly in recent years and provides advantages in ease of management and consolidation, a number of power management techniques specifically address virtualized servers. vGreen tries to improve energy efficiency of virtualized servers by linking workload characterization to dynamic virtual machine (VM) scheduling (Dhiman et al., 2009). Other work studies the power management effectiveness of CPU consolidation on virtualized systems (Hwang et al., 2012). Turning CPU resource limits is a recently introduced power management control knob on virtualized server that can achieve finer granularity power consumption compared to DVFS (Hankendi et al., 2013). However, DVFS settings can be altered more frequently, whereas CPU resource limits can be changed at second level granularities.

2.3.2 Data Center Level Power Management

A data center consists of many servers. In addition to the power management capabilities available within the servers, a data center offers other power management knobs, including power budgeting and server provisioning.

Several power budgeting approaches consider the heterogeneous set of applications and divide total power caps based on application properties (Rajamani et al., 2006; Zhan and Reda, 2013). Gandhi et al. develop a queuing model and produce theorems that determine the optimal power allocation under different scenarios including different arrival rates of jobs, power-to-frequency relationships in the processors, etc (Gandhi et al., 2009). The power budgeting problem has also been studied on virtualized systems (Nathuji et al., 2008; Nathuji et al., 2009).

Server provisioning, which decides how many servers should be active at a given time, is another essential topic in the data center. Many data centers today leave all the unused servers in idle states as a conservative approach for guaranteeing high performance. Leaving many servers idle, however, causes tremendous waste of energy. Some data center researchers leverage sleep states to improve energy efficiency (Chase et al., 2001; Meisner et al., 2011); however, they typically ignore the wake-up costs from sleep states or use hypothetical server states. Gandhi et al. (Gandhi et al., 2012) propose a *SoftReactive* dynamic power management policy, which determines the state of servers in the data center based on the dynamic workload arrival rate, and introduce a timeout-based mechanism to sleep servers.

2.3.3 Workload Scheduling and Control

Data centers serve hundreds of thousands of workloads per day. How to schedule and allocate these workloads to servers impacts both the data center power and QoS performance. A number of scheduling algorithms have been proposed and evaluated. The *first come and first serve* (FCFS) is a simple but popularly used strategy in today's system. The *backfilling* policy is to improve system resource utilization by identifying "holes" in the scheduling plan and moves forward small jobs to fit the "holes" (Mu'alem and Feitelson, 2001). Recently, scheduling algorithms that give small jobs higher priorities are especially of interest. The *shortest job first* is first proposed to always serve the shortest job in the system in a non-preemptive manner. A few of extensions on it, e.g., *shortest remaining processing time* and *preemptive shortest job first* are then proposed, which serve short jobs in the preemptive manner, and are proved to be able to achieve low mean delay and short mean queue length (Yang et al., 2012). *Processor sharing* is a policy that shares service capacity evenly among all requests (Aalto et al., 2007). A multi-class adaption of the processor sharing, i.e., the *generalized processor sharing* is designed to share capacity based on some weight factors to all non-empty classes of workloads (Parekh and Gallager, 1993). Jobs within each class are usually assumed to form a FCFS queue to be served. Generalized processor sharing is suitable for the scenario where data centers are split into multiple clusters, with each cluster serving a type of workload. Processor sharing and generalized processor sharing offer more fairness among workloads than other scheduling algorithms such as the shortest job first.

Data center power is also highly related to the workload arrival rates. When the workload arrival rates are high, a large amount of power is consumed in order to serve the workloads and guarantee QoS. Recent studies propose several workload control methods to regulate workload arrival so as to control the data center power. Widely studied workload control methods include load shedding, shifting and migration. Load shedding is to simply reduce temporary load by turning off servers, without any future pay back. Load shifting is to temporarily turn off servers and reschedule loads to a future spot (Ghatikar, 2014). Load migration is to shift load geographically to other data centers or clusters (Wang et al., 2014). Unlike load shedding and shifting that are usually accompanied with QoS degradation, load migration usually causes less or even no degradation in workload servicing. Load migration also contributes to grid balancing and helps reduce power network congestion. A number of migration strategies and online algorithms have been proposed, and the potential environmental benefits are evaluated (Chiu et al., 2012; Hu et al., 2016; Liu et al., 2011; Lin et al., 2012). VM based migration techniques are also introduced for geographic workload migration (Wang et al., 2013a).

2.4 Data Centers Participation in DR

Studies on data center participation in DR programs have been significantly advanced in recent years. A recent survey provides valuable insight into opportunities and challenges of data center in both legacy and emerging DR programs (Wierman et al., 2014). Kirpes et al. evaluate multiple compensation models for data centers in common DR programs (Kirpes and Klingert, 2016). Real-time dynamic energy pricing (Le et al., 2016), peak shaving (Wang et al., 2012; Govindan et al., 2011) and emergency load reduction (Zhang et al., 2015; Tran et al., 2016; Islam et al., 2016) are three popular legacy DR programs. Many studies investigate data center legacy DR program participation through load shedding, shifting and migration (Ghamkhari and Mohsenian-Rad, 2012; Liu et al., 2014; Wang et al., 2014; Cioara et al., 2016).

There is growing interest in data center participation in emerging DR programs, i.e., capacity reserves. Aikema et al. review multiple types of ancillary service markets for data center to participate, and evaluate the capability and potential benefit (Aikema et al., 2012). Ghasemi-Gol et al., propose an *offline* optimization framework to minimize electric bill of data center in RSR provision (Ghasemi-Gol et al., 2014). Aksanli et al., propose a battery-based design framework for data centers to provide RSR (Aksanli and Rosing, 2014). Li et al. study the joint management of data center and employee plug-in hybrid electric vehicles in RSR provision to further increase the profit (Li et al., 2014). Most of existing studies, however, use simplified data center models for RSR provision, and do not investigate a real-life practical model of data centers that considers heterogeneous workloads, different server power states, their transition delays and energy loss, and workload SLAs, etc. Moreover, most of the studies ignore the optimization problem on power and reserve value bidding (i.e., capacity planning) in their designed optimization frameworks, which is in fact one of the key problems in the overall optimization of data center RSR provision.

2.5 Energy Storage Devices (ESDs) in DR

In addition to data centers, ESDs are considered as promising options for participation in power markets and DR. Today's most popular ESDs include batteries, flywheels (FW), ultra-capacitors (UC) and other emerging techniques, e.g., compress air energy storage (CAES), etc (McCluer and Christin, 2008; Smith et al., 2008). These ESDs are modeled, for either ideal or non-ideal behaviors, and their system performance is evaluated (Wang et al., 2012; Ghiassi-Farrokhfal et al., 2015). Recently, the hybrid electric energy storage system is designed and investigated to enlarge the system storage capacity and improve the efficiency (Pedram et al., 2010).

A few previous studies propose control policies and evaluate the benefit of ESDs in real-time dynamic energy pricing programs (Wang et al., 2013b; Zhu et al., 2013), peak shaving (Wang et al., 2012; Aksanli et al., 2013), power grid stabilization and primary reserve provision (Oudalov et al., 2007; Cho et al., 2013), respectively. In the space of RSR, some prior work surveys potential market chances and evaluates maturity of the ESD participation in RSR (Walawalkar et al., 2007; Kumaraswamy and Cotrone, 2013; Vu et al., 2009; Fooladivanda et al., 2014), but without formulating the detailed models of participation and evaluating the optimal solutions. Kim et al. investigate the optimization solution of ESDs in RSR provision (Kim et al., 2014), however, their study uses a simplified RSR participation model that does not consider the details of regulation accuracy constraints and penalties. Furthermore, it assumes that the RSR signal always follows a statistical distribution known a priori, and without considering the reserve value and capacity planning for different ESDs. Overall, there also lacks a systematic evaluation and comparison on the optimal capabilities and profits from various types of ESDs in different DR programs in literature.

2.6 Distinguishing Aspects from Prior Work

Our work proposes a novel approach to improve the data center energy efficiency and reduce the energy costs through data center participation in emerging DR programs. We specifically investigate an emerging capacity reserve market, the RSR provision, which is a brand new market for data centers. More specifically:

- Rather than reducing the data center power consumption and electricity costs through power management techniques as most prior studies focus on, our research aims to improve data center energy efficiency and reduce the costs through integration of data centers into DR programs in the smart grid. In the programs, the data center also contributes to the power grid stabilization and the integration of renewables into power markets;

- For those studies on data center DR participation, most of them still only focus on legacy programs, such as peak shaving, dynamic energy pricing, or emergency load reduction, while this dissertation targets for an emerging DR program, the RSR provision, which is much more profitable for data centers than those legacy programs;

- Our work is the first to design a practical data center model in RSR provision. By "practical" here, we refer to modeling a wide range of real-life factors in data centers such as the heterogeneities in workload (i.e., different types of applications running), multiple server power states and server provisioning, the associated time delay and energy loss during server state transitions, workload scheduling and allocation, and workload QoS requirements determined by SLAs, etc. The practical model is aware of both data center hardware and software characteristics;

- We design policies based on this practical model. Our policies not only handle those real-life factors and accurately track the RSR signal, but also provide workload QoS guarantees during the RSR provision. We are also the first to not only consider efficient runtime policies, but also the optimal energy and reserve bidding (i.e., capacity planning) to the power market as well in the optimization framework to minimize the data center energy monetary cost with RSR provision;

- We are the first to provide detailed models, evaluate and optimize the profits of various ESD technologies in not only legacy, but also emerging DR, by proposing detailed reserve value and capacity planning as well as online ESD operational policies;

- This dissertation does not merely formulate the data center RSR provision problem and seek to theoretic solutions, but also implements the proposed techniques on a real-life system using existing control knobs in virtualized servers. Such an implementation is the first to provide guidance for future implementation of large-scale data center DR participation systems.

Chapter 3

Data Center Demand Response

3.1 Overview

Today's smart grids incorporate a larger percentage of intermittent renewable energy sources in power generation. These new volatile energy sources create challenges for grid independent system operators (ISOs) to stabilize the grid load and match the power supply with demand in real time. Therefore, ISOs adopt novel mechanisms in modern power markets to ensure stability. Demand response (DR) is one such mechanism, where the demand side participant receives monetary benefits upon regulating its power consumption based on ISO requests.

In tandem with the development in the power markets, electricity used by the data centers has grown to account for 3% of the overall consumption in the US today (Koomey, 2011). Recent advancements in power capping and power management techniques for the servers in the data centers (Li and Martinez, 2006; Isci et al., 2013; Meisner et al., 2009) have enabled the data centers to provide some flexibility in their energy consumption. Therefore, data centers offer a unique opportunity for providing DR. Exploiting this flexibility can help satisfy most of the growth in data center energy consumption from the renewable energy, and also provide additional reserves to other less flexible uses of electricity in our society.

This chapter focuses on evaluating the capabilities and benefits of data center participation in the power market for providing DR. Among a variety of DR pro-

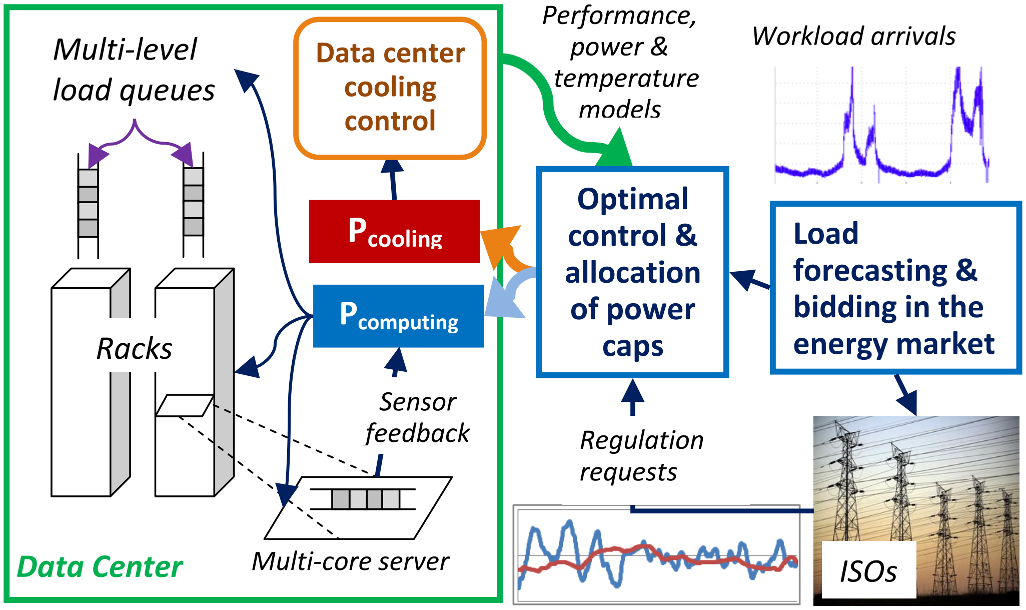

Multi-level load queues
Data center cooling control
Performance, power & temperature models
Workload arrivals
$P_{cooling}$
Optimal control & allocation of power caps
Load forecasting & bidding in the energy market
Racks
$P_{computing}$
Sensor feedback
Regulation requests
Multi-core server
Data Center
ISOs

Figure 3·1: Data center DR participation framework.

grams, the demand side regulation service reserves (RSRs) are especially of interest as their market clearing prices are, on average, as valuable in today's markets as energy clearing prices (PJM, 2013; NYISO, 2016). More importantly, we focus on RSRs because on one hand their requirements are expected to increase rapidly with increasing renewable energy integration in the grid (Makarov et al., 2009), while on the other hand data centers have comparative advantages in offering RSRs relative to other demand side reserve providers.

Figure 3·1 shows how the different sub-components of the data center DR participation problem (i.e., mainly the RSR provision in this dissertation) come together. The whole data center DR participation includes the following steps:

1. The data center first acquires the information of workload arrivals for the next time period (e.g., next hour). If such information is not available, the data center forecasts the workload based on the historical workload patterns;
2. Using an estimation of future workload arrivals and the ISO requirements of the

program that the data center decides to participate in, the data center computes the demand reserves it can provide, and bids in the power market;

3. Once the ISO approves the bid, the request signal is sent to the data center from the ISO. Then the data center optimally distributes the total power cap calculated based on the request signal to the cooling units and to each server. The data center also performs workload allocation to servers. Based on the condition of both the power caps and the workloads, the data center control unit also determines the number of servers that should be turned on, put into sleep or tuned off;
4. The cooling system maintains the thermal constraints and gives temperature feedback to the data center control unit. Because of the larger time constants involved in cooling temperature dynamics, ideally cooling power adjustment is performed less frequently compared to server power regulation;
5. Each server has multiple cores and different levels of load queues. Workloads run on the servers and dynamic power capping is applied to track a given server-level cap. Performance and Quality-of-Service (QoS) feedback from each server is sent back to the control unit;
6. Based on the feedback from the cooling units and servers, the data center re-allocates power caps and workloads, and re-determines the server states, so as to follow the ISO request and to improve performance;
7. The data center repeats the steps above for each time period.

In this dissertation, we focus on the power regulation of the computational units (i.e., servers) for fast regulation, as cooling power can only be regulated as part of slower frequency markets due to the thermal time constants. The overall objective of the problem is to minimize data center energy monetary costs under the constraints of RSR signal tracking and the workload QoS (as determined by users or service level agreements (SLAs)), by designing efficient data center runtime policies, as well as the smart energy and reserve bidding strategies. In this chapter, we first model the data center DR participation problem by introducing the real-life server model, computational unit model and the workload model in Section 3.2. Then in Section 3.3 we conduct an initial study on the RSR provision problem with a single server, which is the main building block of the data center level problem. After that we propose and evaluate three RSR provision runtime policies, namely, the best tracking policy (Section 3.4), the optimal stochastic dynamic programing (DP) policy (Section 3.6), and the EnergyQARE policy (Section 3.7), to dynamically regulate the data center power following the RSR signal, by leveraging server power capping techniques, multiple server power states, and the workload arrangement, etc. With the dynamic policies, we also introduce the optimal energy and reserve bidding strategies for data centers to minimize the energy costs subjected to the signal tracking and workload QoS constraints. To evaluate the benefits from RSR participation, in Section 3.5, we make heuristic comparisons of the energy cost savings from RSR provision (with the best tracking policy) to other energy cost saving strategies. Section 3.8 introduces a real-life implementation of the designed optimization framework and runtime policies on a multi-core server, which provides guidance for the future deployment of the DR participation onto real-life large scale data centers. Since data centers sometimes are thought as special types of large scale energy storage devices (ESDs) in power market and DR (Liu et al., 2014), in order to compare the capabilities and profits of DR participation by data centers to typical ESDs, we investigate performance of different types of ESDs in various DR programs in Section 3.9. Section 3.10 summarizes the chapter.

3.2 The Model of Data Center in DR

A data center[1] system is composed of two major parts: the computational unit and the cooling unit. The computational unit consists of a number of servers[2], and the cooling unit consists fans, computer room air conditioners, water cooling systems, etc. Due to the large thermal time constants involved in temperature dynamics, the adjustment of cooling power is usually less frequent than server power regulation, and thus, the cooling unit may not be as suitable for RSR as the computational unit. In this dissertation, we specifically focus on regulating the computational power. Our technique, however, can be combined with power budgeting techniques (Zhan and Reda, 2013) that distribute a given total power cap into power caps of the subcomponents of the data center, while cooling unit can be regulated to participate slower frequency demand capacity reserve market. Figure 3·2 depicts the overall data center model in DR participation. In the following sections, we first discuss a single server model, followed by the model of the computational unit as a whole. Finally, we introduce the model of the data center workloads.

3.2.1 The Model of Servers

Servers in the data center can be assigned to different states. Typical states include: active, idle, sleep and off (Isci et al., 2013). When a server is running a job, it is "active". We use $P_s(t)$ to denote the power consumption of an active server s at time t. $P_s(t)$ is composed of the dynamic power, $P_{dyn,s}(t)$, and the static power, $P_{static,s}$. The dynamic power changes based on the characteristics of the running job, and can be modulated by power management techniques, such as DVFS (Li and

---

[1] We define data centers in this dissertation broadly including both enterprise data centers and high performance computing (HPC) clusters.

[2] the computational unit also includes networking, storage, uninterruptible power supply (UPS) and other elements. This dissertation focuses on providing RSR using server-level controls.

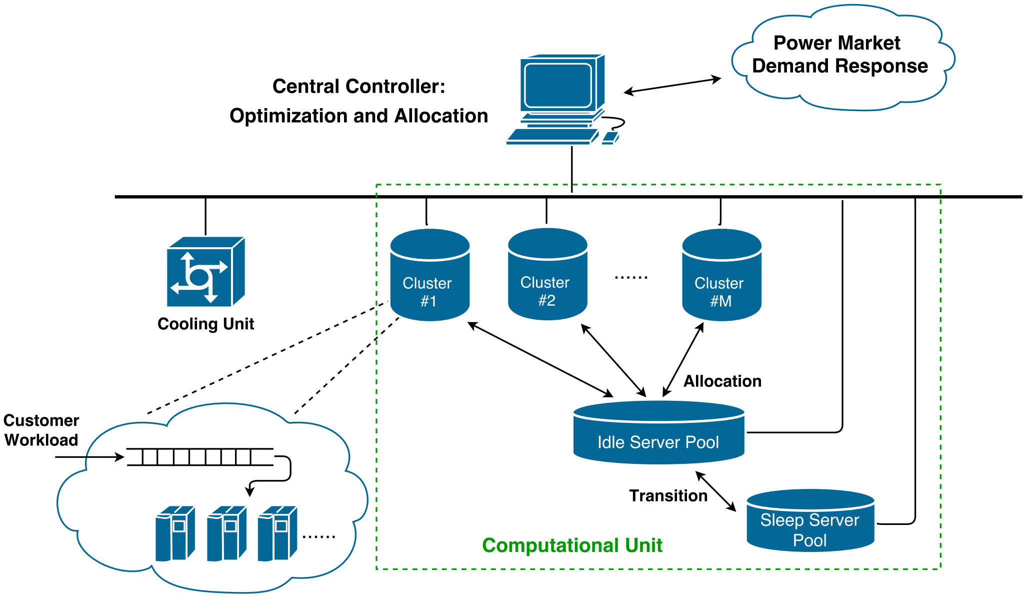


Figure 3·2: The model of data center in DR participation.

Martinez, 2006), CPU resource limits (Hankendi et al., 2013), etc. The static power is a constant[3], and exists as long as the server is turned on.

Regulating the dynamic power affects the server throughput (i.e., service rate). Prior studies have demonstrated a linear relation between dynamic power and the server throughput for active servers (Dayarathna et al., 2016) as:

$$P_{dyn,s}(t) = k_j \cdot u_s(t), \tag{3.1}$$

where $u_s(t)$ is the server throughput of the server s at time t, k_j is a constant which is specific to the type of the job j that is serviced in the server s at time t.

To examine this model, we conduct our experiments on a 1U server that has an AMD Magny Cours (Opteron 6172) processor, which has 12 processing cores on a single chip. The server is virtualized by the VMware vSphere 5.1 ESXi hypervisor. We use the *CPU resource limits* control knob in the hypervisor to control the power-

---

[3] The static power, in fact, is temperature dependent. We assume that there is no temperature change in this dissertation.

performance settings at runtime (Hankendi et al., 2013). CPU resource limits enable dynamically changing the resources allocated to a virtual machine (VM) quickly and at a fine granularity. For example, cutting the resource limits for a VM to half of the original setting (without limits) cuts the server's power consumption also to half of the original power level while running that VM. Similarly, we can set the performance of the server to any level we need. As more than 50% of the data center servers are virtualized today, controlling the power and performance for the applications through changing the resource limits for the VM is a practical and efficient method.

We run each application from the PARSEC-2.1 (Christian, 2011) benchmark suite on a VM in isolation (by itself, without consolidation) in our experiments and apply regression on the data collected to derive the model between the server power consumption $P_s(t)$ and the server throughput $u_s(t)$. The server throughput in the experiments is represented by the Retired Instructions Per Second (RIPS). A job running on a server is composed of a number of instructions. Finishing a job is equivalent to executing all its instructions. RIPS is a metric showing the number of instructions finished in each second, and is commonly used for evaluating the performance of the processor. A higher RIPS represents a faster processor service rate. We construct the following model with a mean square error of less than 5%:

$$P_s(t) = k_j \cdot u_s(t) + P_{static,s}, \tag{3.2}$$

which matches with the classical used model introduced in Eq. (3.1). The data and model fits are shown in Figure 3·3.

A server is "idle" if it is turned on but is not running any jobs. An idle server consumes power at a constant rate, $P_{idle,s}$, which is equal to $P_{static,s}$. In the "sleep" state, the server consumes a very low constant power, $P_{slp,s}$. We assume to have homogeneous servers in our study, and thus we omit the notation s in those constant

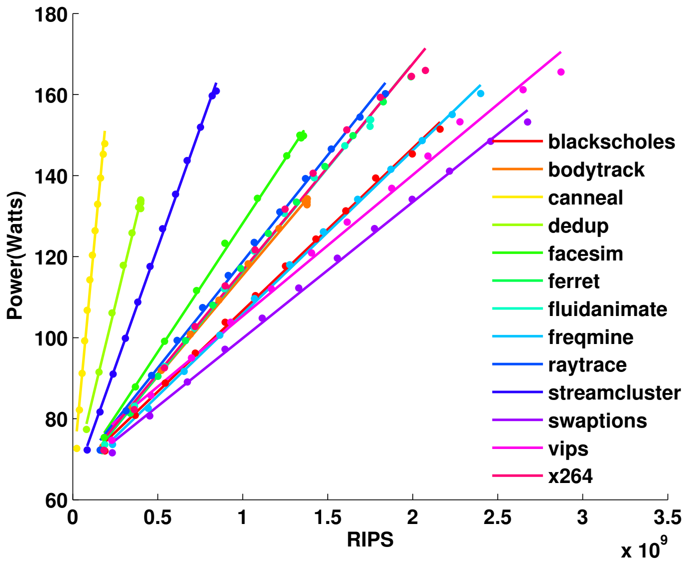


Figure 3·3: The relation between server power and throughput (in RIPS) for applications in PARSEC-2.1 benchmark suite. Dots are the real measurements and lines are the linear fitting curves.

values for simplicity in the rest of the dissertation. There are time delays and energy loss of resuming a server from or suspending it to a "sleep" state. The suspending time delay, T_{susp}, is usually small and can be ignored, while the resuming time delay, T_{res}, is notable and requires explicit consideration (Gandhi et al., 2012; Isci et al., 2013). During both the suspending and resuming periods, the power consumption is similar and denoted as P_{tran}, which is close to the maximal server power, P_{max} (Isci et al., 2013). The energy loss of the resuming period is estimated as $E_{loss} = T_{res} \cdot P_{tran}$ and of the suspending period can be ignored.

Many servers in today's data centers are able to be set into different sleep states. In this dissertation without loss of generality, we study two types of sleep states: the shallow sleep state and the deep sleep state. Servers in the deep sleep state can save more power, whereas the time delay and the energy loss of them in the rebooting process are larger than those of servers using shallow sleep state. Based on recent studies (Gandhi et al., 2012; Isci et al., 2013), in our work we assume parameters as

Table 3.1: Descriptions of symbols on server power states.

Symbol	**Description**
E_{loss}	Energy loss during the server resuming process.
$P_s(t)$	The power of active server s at time t.
$P_{dyn,s}(t)$	The dynamic power of active server s at time t.
P_{static}	Static power of server.
P_{idle}	Server power in the idle state.
P_{tran}	Server power during the transition state.
P_{slp}	Server power in the sleep state.
P_{max}	The maximal possible server power.
T_{susp}	Suspending time delay to the sleep state.
T_{res}	Resuming time delay from the sleep state.
$u_s(t)$	Server throughput (i.e., service rate) of the server s at time t.

follows: the sleep power in the shallow and deep sleep states is $P_{slp}^S = 10\% P_{max}$ and $P_{slp}^D = 5\% P_{max}$, and the resuming time is $T_{res}^S = 10s$ and $T_{res}^D = 200s$, respectively. During the transition process, servers consume constant power: $P_{tran}^S = P_{tran}^D = P_{max}$.

Servers in data centers can be completely turned off, which indicates a fourth state, "off", with no power consumption. However, the "off" state does not frequently appear due to the very large time delays and energy loss of resuming and suspending process. Thus, we do not consider the "off" state in this dissertation. Table 3.1 lists the descriptions of all the major symbols introduced in this section on the server power states.

3.2.2 The Model of the Computational Unit

As shown in Figure 3·2, in our data center model, servers in the computational unit are classified into several sub-units based on their states and the types of workloads that they are serving. All the idle servers are assigned into the idle server pool, and all the sleeping servers are in the sleeping server pool. At runtime, active servers are dynamically classified into several clusters depending on the workload types they are serving, with each cluster containing servers that specifically serve one type of workload. In this way, though the overall data center receives heterogeneous workloads, each cluster always serves homogeneous workload. This model is, in principle, similar to the design of today's high performance computing (HPC) clusters, where dedicated and optimized sets of servers are assigned to specific jobs. For example, if there are M different types of workloads in the data center, then the data center contains M active server clusters, one idle server pool, and one sleeping server pool at that moment. $M = 1$ if the data center only serves homogeneous workload. All the active server clusters share the common idle server pool, i.e., idle servers in the pool are dynamically assigned by the central controller to clusters as needed. Active servers are immediately released back to the idle server pool from the clusters if they finish their jobs. In other words, servers in active clusters are all active. The number of servers in each active server cluster, as well as the power budget for each cluster, are dynamically modulated by the central controller based on multiple system states such as the length of the job waiting queue in each cluster, the value of the RSR signal and the QoS constraint of each workload type, etc. The sleeping server pool only interacts with the idle server pool. As introduced in Section 3.2.1, we ignore the time delay of suspending a server from idle to sleep state. Thus idle servers that are put into sleep are immediately moved to the sleeping server pool. On the other hand, if sleeping servers are resumed, they are first in the transition state for a certain time before they become idle and join in the idle server pool.

In each active server cluster, we assume there is a *first come first serve* (FCFS) queue for holding the incoming jobs submitted by the users. Once a job arrives, it is first put into the queue and waits to be scheduled for service. We use the FCFS queue because it is simple but efficient, and is one of the most widely used scheduling policies in today's systems. Moreover, as each cluster contains only homogeneous workload, other scheduling policies such as *shortest job first* and *shortest remaining* *processing time* (Yang et al., 2012) do not provide additional benefits. In addition, we assume that each server can only serve one job a time; thus, we do not consider server consolidation, which is the typical case in HPC data centers. We also assume that a running job cannot be stalled or preempted, to preserve both job QoS and fairness in the service.

Utilization is an important parameter to describe how busy the data center (computational unit) is. It is defined as the average number of active servers at each time interval. For example, $U = 50\%$ means each server is active for half of the whole period, and is in idle or sleep state for the rest of the time. We can also comprehend this as, at each moment, half of total servers in the data center are serving jobs. The utilization depends on the arrival frequency and the servicing time of workloads.

3.2.3 The Model of Data Center Workloads

Workloads in data centers mainly fall into two catalogs: (1) interactive jobs such as email clients, web search, stock transactions, etc., which are highly sensitive to the latency, and (2) batch jobs that are more tolerable to latency. Some of the batch jobs can be accumulated and run later when there is availability in the data center (Liu et al., 2012; Verma et al., 2015). Therefore, batch jobs provide additional flexibilities in data center power consumption, and clusters with batch jobs are more suitable for DR participation. In our work, we focus on DR provision on data center clusters that mainly serve batch jobs. In a general data center scenario with a mixture of both interactive and batch jobs, it is possible to enable that clusters with interactive jobs mainly focus on workload servicing, and provide only a few to none reserves, while clusters with batch jobs actively participate in DR programs.

In data center, the *system time* $T_{sys,i}$ (i.e., waiting time plus processing time) is one of the most significant QoS indexes for each job i. Different types of workloads, however, can have very different processing time, which makes the simple comparison of system time $T_{sys,i}$ among different workloads unfair. To unify the QoS among all types of workloads, we normalize the system time $T_{sys,i}$ to the shortest processing time for the job i, i.e., $T_{min,i}$, which refers to running the job without any power capping restrictions and without any waiting time in the queue. $T_{min,i}$ is a constant for a given job i and can be known ahead by profiling the workload. Hence, $T_i = T_{sys,i}/T_{min,i}$, where T_i is the normalized system time of job i. Sometimes the QoS degradation D_i, i.e, $D_i = T_i - 1$, is also used instead of T_i. T_i and D_i are basically equivalent QoS measures. $D_i = 1$ means that there is no QoS degradation (i.e., the job is finished in the shortest possible time).

We then propose the service level agreement (SLA) based on D_i (or T_i) in our study. SLAs today are mainly defined as the availability of the service, e.g., 99.9% of the time the service is guaranteed to be available (Amazon, 2013). However, customers and service operators start to include performance measures of the service, e.g., the throughput, or the delay of service, into SLAs (Ghamkhari and Mohsenian-Rad, 2012). In our work, we propose to design SLAs based on the job system time (i.e., based on the QoS degradation), as it is one of the key measures to evaluate the job servicing performance. We further propose to design SLAs in a probabilistic form, which is suitable for more general uses. Since each active server cluster j serves homogeneous workload, we use parameters $(Q_j,\ \eta_j)$ to characterize the SLA for the workload in cluster j as:

$$Probability \left\{ D_{i,j} \leq Q_j \right\} \geq \eta_j, \tag{3.3}$$

which represents that the job i in cluster j is required to be served within a certain threshold of QoS degradation, i.e., Q_j, with a probability larger than η_j.

3.3 An Initial Study: Single Server RSR Provision

We start to focus on problems of RSR provision. In this chapter, we first conduct an initial study on the RSR provision with a single server, which constitutes the main building block of the overall framework of the data center in Figure 3·2, to provide the proof-of-concept for capabilities and benefits of a data center in RSR provision.

3.3.1 Runtime Policy and Optimal Bidding

In the single server scenario, we do not need to solve any power budgeting or server provisioning problems (we assume the only server is never put to sleep). To better mimic the data center scenario, we assume multiple workload queues with different servicing priorities. We use a simple runtime policy for the single server RSR provision as follows:

1. Upon completion of servicing a job, the server selects a new job from the highest priority queue that is not empty, using a FCFS protocol;
2. The server has a range of power consumption rates for that job. The policy, in real time, selects an allowable consumption rate that minimizes the instantaneous tracking error $\epsilon(t)$.

Next, we formulate the optimization problem to solve the optimal bidding values of energy and reserve, i.e., $(\bar{P},\ R)$. The optimal bid should minimize the electricity monetary costs while meeting the ISO requirements and user SLA constraints.

We study the sample frequency distributions of the tracking error $\epsilon(t)$ and the degradation D_j^p (p, j here denote the priority and workload type, respectively) trajectories for a sufficiently large number of simulations, and notice that they mostly fit the Gamma distribution, Γ, with parameter shape k: $k_\epsilon = \bar{\epsilon}^2/\sigma_\epsilon^2$, $k_{D_j^p} = \bar{D_j^p}^2/\sigma_{D_j^p}^2$ and scale θ: $\theta_\epsilon = \sigma_\epsilon^2/\bar{\epsilon}$, $\theta_{D_j^p} = \sigma_{D_j^p}^2/\bar{D_j^p}$, for $\epsilon(t)$ and D_j^p, where $\bar{\epsilon}$, σ_ϵ, $\bar{D_j^p}$, $\sigma_{D_j^p}$ are means and standard deviations of $\epsilon(t)$ and D_j^p. Hence when we solve the optimal bidding problem, we use Gamma distribution to construct the probabilistic constraints. The parameters of tracking error probabilistic constraints are $(\epsilon^{tol}, \eta^{\epsilon})$, and of SLAs are (Q_j^p, η_j^p), respectively as introduced before.

Finally, we apply limits on dynamic power consumption based on the maximal achievable power value P_{max}, and the server idle power P_{idle} that is basically the minimal achievable power value, assuming that the server is never put to sleep or turned off. The optimization problem is formulated as follows:

$$\begin{aligned} \underset{\bar{P},R,\ \text{policy}}{\text{minimize}} \quad & \Pi^E \bar{P} - (\Pi^R R - \Pi^{\epsilon} \cdot \bar{\epsilon}) \\ \text{subject to} \quad & \Gamma(k_{\epsilon}, \theta_{\epsilon}, \epsilon^{tol}) \geq \eta^{\epsilon}, \\ & \Gamma(k_{D_j^p}, \theta_{D_j^p}, Q_j^p) \geq \eta_j^p, \\ & \bar{P} + R \leq P_{max}, \\ & \bar{P} - R \geq P_{idle}, \\ & \bar{P} \geq 0, R \geq 0, \end{aligned} \tag{3.4}$$

where Π^E is the hour ahead clearing price of energy and Π^R is the hour ahead clearing price of reserve, both in \$/kWh, introduced in Section 2.2. Today, the power market has $\Pi^R \approx \Pi^E$ for RSR (PJM, 2016). Π^{ϵ} is the penalty price on the signal tracking error.

We design a bidding engine to calculate the optimal bid. The electricity price information (Π^E, Π^R and Π^{ϵ}), a sample RSR signal $y(t)$, the tracking error tolerance given by the ISO (ϵ^{tol}, η^{ϵ}) and the server specific information, e.g., P_{idle} are saved in the engine. The inputs of the engine are the information on the workloads and customer SLAs for the next hour. Using the workload information, a power-throughput model (see Section 3.2.1) is derived first. Then along with the power-throughput model, all these inputs are sent to a simulator that simulates the whole RSR provision process. The simulator uses exhaustive search to find the optimal ($\bar{P}$, R) values that satisfy the constraints. It is possible to first conduct a sensitivity analysis on $\bar{P}$ and R and use the results to construct a more structure search. Using exhaustive search, simulation takes only a few seconds; so it is not necessary to optimize the search for the problem size we focus.

We assume that workload information (i.e., workload types and arrival rates) for the following hour is provided in advance in our study. This is reasonable as for many real-life cases in the data centers, information of workloads is provided by customers to the data center some time before they start executing (e.g., in the case of batch job submissions in HPC clusters). Mechanisms for workload forecasting can be also designed and used in conjunction to our optimization technique.

3.3.2 Experimental Results

In our experiments, without loss of generality, we assume that all jobs are classified in two priority levels: high and low. Thus, we have one *high priority queue* (HPQ) and one *low priority queue* (LPQ). We assume that jobs arrive to the server following a Poisson process, and we set job arrival rates to achieve a system utilization around 50%, which is typical in today's data centers. Jobs with different priorities arrive at different arrival rates; i.e., higher priority jobs have a lower arrival rate, as in general the number of higher priority jobs will be smaller than that of the low priority jobs. Similarly, high priority jobs are more urgent in general, and hence operate under tighter SLA constraints. We assume that the arrival rate of the low priority jobs is three times larger than that of high priority jobs. We generate the job queues using Monte Carlo simulation. Finally, in order to measure statistics of tracking error and QoS, we simulate a 1-hour period 10 times to achieve statistical confidence.

Tracking Performance and QoS Evaluation

We first investigate the performance under the circumstance of homogeneous workload but with different priorities. Figure 3·4 shows the performance of signal tracking along with HPQ and LPQ QoS degradation of the jobs *Blackscholes*. We see from the signal tracking figure that at the given near-optimal $(\bar{P}, R)$ setting and under the policy proposed in Section 3.3.1, our system is able to track the RSR signal with small errors in most time. Larger errors only appear when both queues are empty, in which case the server is forced to stay in the idle state with power consumption of P_{idle} and cannot be regulated. The QoS degradation figures show that the degradation of LPQ is much larger than that of HPQ. This is because LPQ has a larger arrival rate and it is always served after the HPQ jobs are served. Experiments on other workloads show similar results in both signal tracking and QoS performance, which imply that providing RSR is not constrained by the job type.

We next investigate a heterogeneous case; i.e., jobs arriving at the server are of different types. Without loss of generality, we assume all the jobs are either Blackscholes or Canneal. Figure 3·5 shows the signal tracking and the QoS degradation of each job type separately at $\bar{P} = 115.33W$ and $R = 30W$ (i.e., a near-optimal setting). The result has limited differences compared to the homogeneous case, which implies that RSR can be provided for a set of heterogeneous jobs arriving at the system.

Optimal Solution and Monetary Savings

Then we study the optimal solution of the single server RSR provision and estimate the corresponding data center electricity monetary savings. The objective function in Eq. (3.4) is based on the monetary costs for *a single server per hour*, when the server consumes power at an average level $\bar{P}$ W and provides RSR of R W. A data center generally contains thousands of servers. Table 3.2 shows the electricity mone-

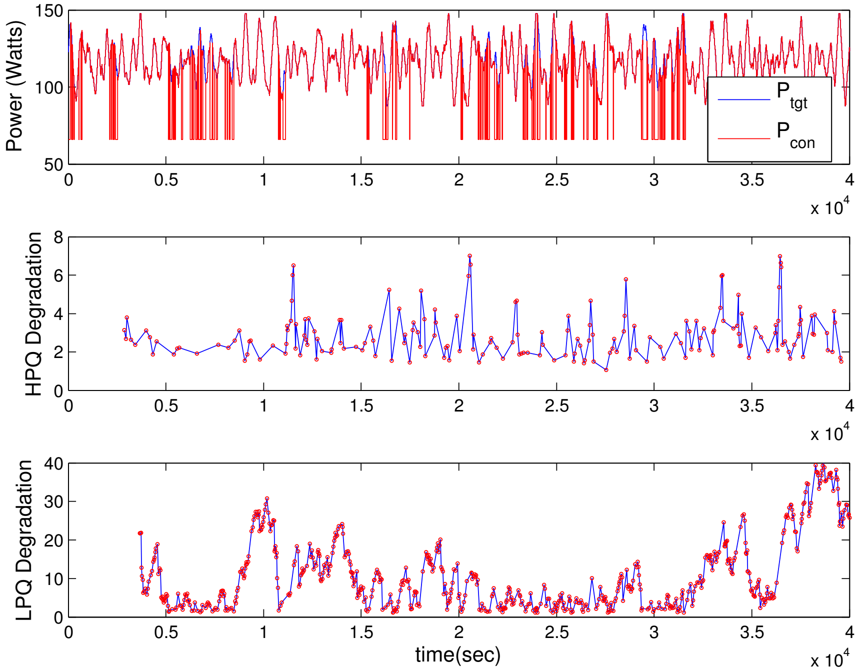


Figure 3·4: Results of Blackscholes with $\bar{P} = 117.65W$ and $R = 30W$ in single server RSR provision. (i) $P_{tgt}(t) = \bar{P} + y(t)R$ and $P_{con}(t)$ trajectories (in Watts) over a 11-hour period (10 replications of a 1-hour period, the first hour data is not used because of the warming up process). The tracking error statistics: $\bar{\epsilon} = 0.20$, $\sigma_{\epsilon} = 0.60$. (ii) HPQ QoS degradation for each job arrival shown as a red dot on the time trajectory. The overall statistics: $\bar{D}^{H}_{bls} = 2.91$, $\sigma_{D^{H}_{bls}} = 1.22$. (iii) LPQ QoS degradation for each job arrival shown as a red dot on the time trajectory. The overall statistics: $\bar{D}^{L}_{bls} = 11.27$, $\sigma_{D^{L}_{bls}} = 9.31$.

tary costs ($/h) for a data center that has $10,000$ servers of the same type running both homogeneous and heterogeneous job cases at various ($\bar{P}$, R) values. The bold highlighted line is the optimal solution of the Eq. (3.4) solved by brute force method at a sufficiently fine granularity. In solving Eq. (3.4), the following parameters are used: $\Pi^R = \Pi^E = \Pi^{\epsilon} = 0.1\$/kWh$, $\eta^{\epsilon} = \eta^{p}_{j} = 0.85$, $\forall p, j$, $\epsilon^{tol} = 0.2$, $P_{idle} = 66W$, $Q^{H}_{j} = 5$ and $Q^{L}_{j} = 25$, $\forall j$. P_{max} changes between 130W and 170W depending on the job type.

The results show that in all cases, the solution is optimal when the reserve R is

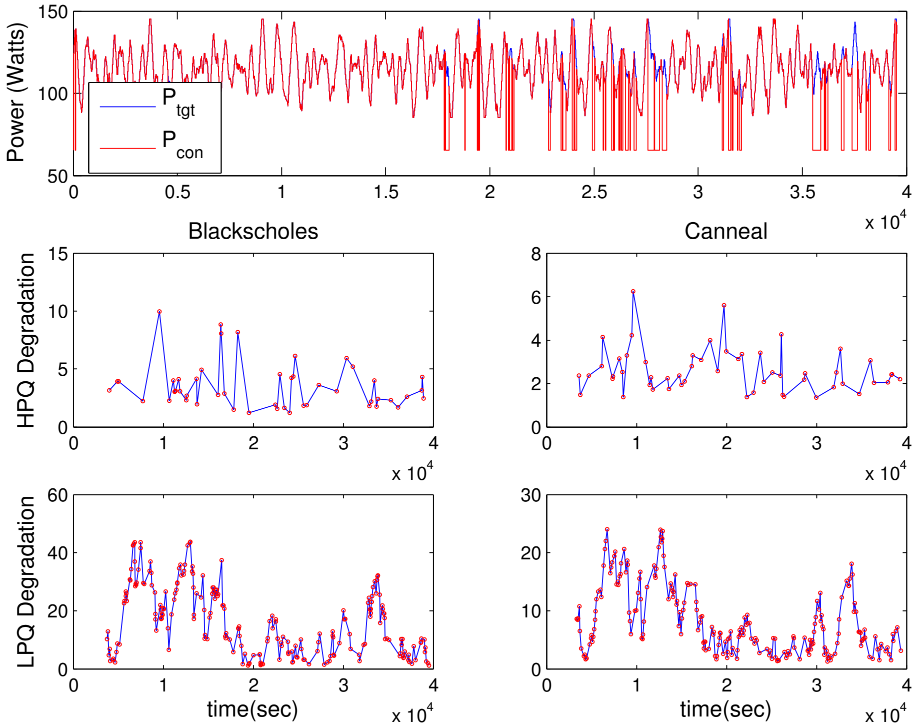


Figure 3·5: Results of mixed workload of Blackscholes and Canneal in single server RSR provision with $\bar{P} = 115.33W$ and $R = 30W$. (i) $P_{tgt}(t) = \bar{P} + y(t)R$ and $P_{con}(t)$ trajectories (in Watts). The tracking error statistics: $\bar{\epsilon} = 0.20$, $\sigma_{\epsilon} = 0.58$. (ii) HPQ QoS degradation trajectory with overall statistics: for Blackscholes $\bar{D}^H_{bls} = 3.53$, $\sigma_{D^H_{bls}} = 2.02$ and for Canneal $\bar{D}^H_{can} = 2.64$, $\sigma_{D^H_{can}} = 1.01$. (iii) LPQ QoS degradation trajectory with overall statistics: for Blackscholes $\bar{D}^L_{bls} = 16.84$, $\sigma_{D^L_{bls}} = 11.63$ and for Canneal $\bar{D}^L_{can} = 8.36$, $\sigma_{D^L_{can}} = 5.77$.

around 30% of its corresponding $\bar{P}$, and 23% of the P_{max} (P_{max} of each job type is shown in the bottom row of the table). Such a result also implies that the optimal percentage of RSR provision does not change much among different types of jobs. In addition, comparing the monetary costs under the optimal solution ($\bar{P}$, R) to those in the first row of the table, which do not have any RSR provision (i.e., R=0), we see that the monetary savings are approximately 30%, which is highly promising. Note that 'N/A' in the table means that there is no feasible solution for the corresponding ($\bar{P}$, R) pair according to Eq. (3.4).

Table 3.2: The electricity monetary costs via different workload types and ($\bar{P}$, R), in single server RSR provision.

Blackscholes			Bodytrack			Canneal			Facesim			Streamcluster			Blackscholes + Canneal		
$\bar{P}$	R	Cost	$\bar{P}$	R	Cost	$\bar{P}$	R	Cost	$\bar{P}$	R	Cost	$\bar{P}$	R	Cost	$\bar{P}$	R	Cost
117.65	0	117.65	103.63	0	103.63	113	0	113.00	115.04	0	115.04	117.65	0	117.65	115.33	0	115.33
117.65	10	N/A	103.63	10	94.36	113	10	104.36	115.04	10	N/A	117.65	10	N/A	115.33	10	N/A
117.65	20	98.45	103.63	20	83.83	113	20	93.35	115.04	20	95.48	117.65	20	N/A	115.33	20	96.12
117.65	30	88.03	103.63	30	73.73	113	30	83.17	115.04	30	85.24	117.65	30	87.91	115.33	30	85.70
117.65	**35**	**82.94**	**103.63**	**31**	**72.72**	**113**	**33**	**80.14**	**115.04**	**34**	**81.20**	**117.65**	**35**	**82.84**	**115.33**	**34**	**81.63**
117.65	40	N/A	103.63	40	N/A	113	40	N/A	115.04	40	N/A	117.65	40	N/A	115.33	40	N/A
152.95	0	152.95	134.72	0	134.72	146.9	0	146.90	149.55	0	149.55	152.94	0	152.94	149.92	0	149.92

Table 3.3 shows the comparison in QoS degradation statistics and monetary costs between the case of optimal RSR provision and the case of provision without power regulation (Non-reg.) for different job types. We see that the QoS values in these two cases are very close. Thus, we do not sacrifice much QoS, while we are able to save 30% monetary costs by providing RSR.

Sensitivity Analysis

In real-life data centers, the QoS requirements and the utilization (job arrival rates) of the system frequently change. As a result, the optimal operating point ($\bar{P}$, R) needs to be adjusted. Sensitivity analysis studies how tracking error and QoS degradation vary if ($\bar{P}$, R) changes, and provides information on which direction to search for the new optimal point. Thus, sensitivity analysis can highly improve the efficiency of the brute force method. Many approaches have been proposed for performing sensitivity analysis. In our work, we use the *Finite Difference* (Maly and Petzold, 1996) method.

We conduct sensitive analysis experiment for the homogeneous case with Blackscholes, and measure the changes of tracking error, HPQ and LPQ QoS degradation

Table 3.3: Performance comparison of optimal RSR provision and provision without regulation on single server.

		$\bar{D}_j^H$	$\sigma_{D_j^H}$	$\bar{D}_j^L$	$\sigma_{D_j^L}$	Cost($/h)
Blackscholes (117.65, 35/0)	Optimal:	3.01	1.40	11.58	9.60	82.94
	Non-Reg:	2.73	0.99	10.92	9.31	117.65
Bodytrack (103.63, 31/0)	Optimal:	3.27	1.27	13.65	7.22	72.72
	Non-Reg:	2.97	0.83	12.37	7.01	103.63
Canneal (113, 33/0)	Optimal:	3.06	1.35	13.17	7.35	80.14
	Non-Reg:	2.64	0.76	12.30	6.97	113.00
Facesim (115.04, 34/0)	Optimal:	2.53	0.70	7.00	3.53	81.20
	Non-Reg:	2.71	0.63	7.11	3.44	115.04
Streamcluster (117.65, 35/0)	Optimal:	2.33	0.62	6.43	3.27	82.84
	Non-Reg:	2.46	0.55	6.52	3.18	117.65
Blackscholes + Canneal (115.33, 34/0)	Optimal, Bls:	3.73	2.20	16.94	11.70	81.63
	Optimal, Can:	2.72	1.05	8.44	5.79	
	Non-Reg, Bls:	3.18	1.16	17.17	12.09	115.33
	Non-Reg, Can:	2.42	0.57	8.59	5.92	

statistics when either $\bar{P}$ or R is increased by 1%. The results show that while increasing $\bar{P}$ by 1%, first, tracking error increases. This is because higher $\bar{P}$ increases the idle time of the system, in which the system power cannot be regulated. Secondly, the LPQ QoS degradation highly decreases, but the HPQ QoS degradation has no notable change. As expected, increasing $\bar{P}$ leads to QoS improvement, especially for LPQ that has lower priority in job servicing. HPQ jobs are always given priorities for execution, hence the improvement in their QoS is limited. On the other hand, while increasing R by 1%, neither tracking performance nor QoS have notable changes. Such results show that both tracking performance and QoS are more sensitive to $\bar{P}$ than R. Therefore, when searching for the new optimal ($\bar{P}$, R), determining $\bar{P}$ based on new system restrictions and requirements is necessary before selecting R.

3.4 The Best Tracking Runtime Policy

Now we study the RSR provision of the data center. The RSR provision problem on the data center is more complex than that of a single server, as many additional factors are required to be considered, such as power budgeting among servers, workload allocation, and server provisioning, etc. In this section, we propose a *best tracking* runtime policy, the first policy that handles data center level RSR provision. The best tracking policy leverages multiple server power states to minimize the instantaneous tracking error, while also reducing energy waste and avoiding job stalling in the systems. The policy is suitable for a scenario when high signal tracking accuracy is required with relatively loose workload QoS constraints. As a starting point, we study the policy with a homogeneous set of workloads. The policy, however, is able to be easily extended to heterogeneous workload scenarios utilizing existing power budgeting techniques (Zhan and Reda, 2013; Nathuji et al., 2008).

3.4.1 Policy Details

The main idea of the best tracking policy is to modulate the data center power consumption $P_{con}(t)$ to track the RSR signal power cap $P_{tgt}(t) = \bar{P} + y(t)R$ as accurate as possible, under some basic rules on workload QoS guarantee and energy conservation. The available controls for modulating the data center power consumption in the policy include: (a) regulating power consumption of active servers; (b) resuming sleeping servers; (c) suspending idle servers to sleep; (d) activating idle servers with queued jobs. The basic rules on workload QoS guarantee and energy conservation are designed as follows:

1. Running jobs are non-preemptive, and must be served at the power with a minimal bound P_{min}. This rule is designed to avoid jobs being stalled in the system. P_{min} can be determined by (Q_j, η_j) in SLAs;

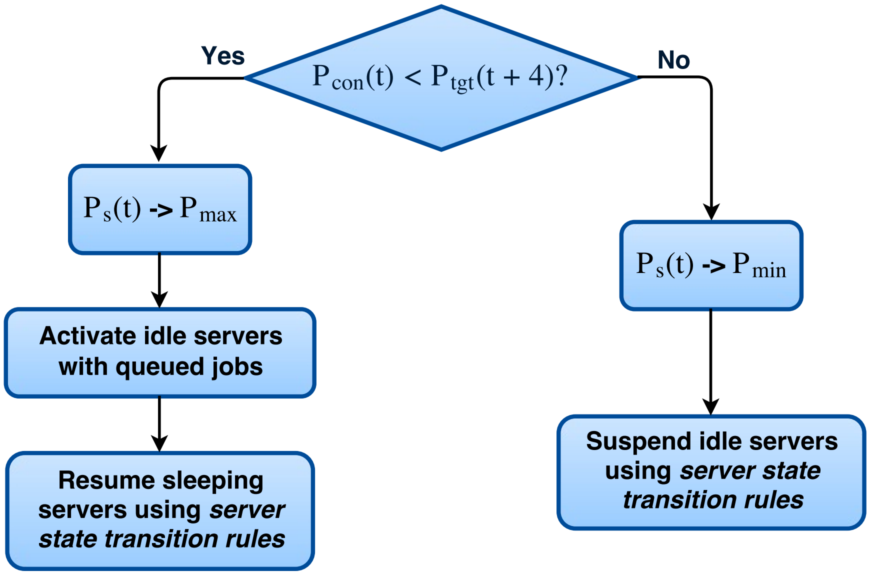

Yes
$P_{con}(t) < P_{tgt}(t+4)$?
No
$P_s(t)$ -> P_{max}
Activate idle servers with queued jobs
Resume sleeping servers using *server state transition rules*
$P_s(t)$ -> P_{min}
Suspend idle servers using *server state transition rules*

Figure 3·6: The flowchart of the best tracking policy.

2. Server state transition rules: if and only if a server s has been in idle for $T_{idle,s}(t)$ that is longer than a timeout threshold, i.e., T_{out}, then it automatically goes to sleep for energy conservation. This timeout mechanism is also designed to avoid over-frequent server transitions, which may cause tremendous energy waste. The threshold is determined based on prior work (Gandhi et al., 2012):

   $$T_{out} = \frac{P_{tran} \cdot T_{res}}{P_{idle}}. \tag{3.5}$$

   In addition, in order to maximize the number of sleeping servers to save energy, idle servers with the smallest $T_{idle,s}(t)$ are always first selected to be activated and serve the queued jobs. Similarly, if some servers are required to be put into sleep, servers with the largest $T_{idle,s}(t)$ are selected at first.

Since the best tracking policy sets signal tracking as the highest priority goal, the main state used for decision making is the dynamic signal power $P_{tgt}(t)$. The flowchart of the best tracking policy is in Figure 3·6. We use $q(t)$ and $N_{idle}(t)$ to denote the total number of jobs in the queue and the number of idle servers at time t, respectively, then the policy is as follows:

Case 1- If $P_{con}(t) < P_{tgt}(t+4)$, i.e., the power consumption needs to be increased to meet the signal, then we do the following three steps in the order until $P_{con}(t)$ meets the power cap $P_{tgt}(t+4)$:

1. Increase power consumption $P_s(t)$ of some active servers s that are not running at maximal capability to P_{max};
2. If $q(t) > 0$ and $N_{idle}(t) > 0$, then activate some idle servers and run them at maximal capability with power consumption at P_{max};
3. Resume sleeping servers following the server state transition rules.

Case 2- If $P_{con}(t) > P_{tgt}(t+4)$, i.e., the power consumption needs to be decreased, then we do the following three steps in the order until $P_{con}(t)$ meets the power cap $P_{tgt}(t+4)$:

1. Decrease power consumption $P_s(t)$ of some active servers s that are *not* running at maximal capability to P_{min};
2. Decrease power consumption $P_s(t)$ of some active servers s that are running at maximal capability to P_{min};
3. Suspend idle servers to sleep state following the server state transition rules.

Note that in the policy we attempt to maximize the number of servers that run at their maximal capability in order to save energy. This is because that the relation between power consumption and server throughout is linear as introduced in Section 3.2.1. Setting the server at its maximal throughput to reduce the processing time helps minimize the energy waste caused by the server static power P_{static}.

3.4.2 Energy and Reserve Bidding

Since in the best tracking policy we assume to have loose SLA constraints, the optimization problem in Eq. (3.4) can be simplified. Instead of solving the optimal energy and reserve bidding with simulations and exhaustive search, in this section, we propose a way to directly estimate an analytical solution of proper bidding values.

Similar to previous settings, we assume the arrival of workloads is a Poisson process with an arrival rate λ (per hour). The value of λ can be controlled by allocating overall load among geographically dispersed data centers to exploit spatiotemporal variations in energy prices (Wang et al., 2014). The λ considered here is the one after such allocation. Each job i is composed of a number of instructions, namely, I_i. Since we assume the homogeneous workload, then all $I_i,\ i = 1, 2...$ are equal and denoted as I. Finishing a job is equivalent to executing all the instructions.

Having λ and I, we can estimate the average number of required active servers, $\bar{N}_{act}$, in order to finish all jobs during the hour. Since our designed policy tries to always enable active servers running at their maximal capability, then $\bar{N}_{act} = \frac{\lambda \cdot I}{u_{max}}$, where u_{max} is the maximal available server throughput. Then the average power consumption from the active servers is estimated as $\bar{N}_{act} \cdot P_{max}$.

Next, we estimate the energy loss during transition periods. As introduced before, each resuming process has an energy loss as E_{loss}. Assuming the total number of times that servers are resumed during the hour is N^h_{res}, then the total energy loss during the hour is: $E^h_{loss} = E_{loss} \cdot N^h_{res}$. We estimate N^h_{res} as follows: since the dynamic range of RSR signal $y(t)$ is $[-1, 1]$, and we use the best tracking policy, when $y(t) = -1$, the data center ideally should be at its lowest power consumption, P_{low}, and when $y(t) = 1$, the data center should be at its highest power consumption, P_{high}. In order to maximize the possible reserve value R so that the reserve credits can be maximized, $P_{high} - P_{low}$ should be maximized, i.e., P_{low} should be minimized and P_{high} should be maximized. The ideal minimal P_{low} that can be achieved is to put all servers to sleep, and the ideal maximal P_{high} is to active all servers with their maximal power. Thus, every time RSR signal increases from -1 to 1, all the servers in the data center are resumed at least once. On the other hand, since our designed policy applies the idle server timeout mechanism to prevent servers from being resumed and suspended back and forth, most of server resuming processes only happen when the power signal has a large increase. We denote the times of the large increases in power signal value during the hour as ψ, and estimate N^h_{res} as $N^h_{res} = \psi \cdot N$, where N is the total number of servers in the data center. Based on the observation of the signal pattern, a reasonable estimation of ψ is $\psi = 3 \sim 4$.

Now we estimate the average power consumption $\bar{P}$ as follows:

$$\bar{P} = \frac{\int_0^{1h} (\bar{P} + Ry(t))dt}{1h} = \bar{N}_{act} \cdot P_{max} + \bar{N}_{idle} \cdot P_{idle} + \bar{N}_{slp} \cdot P_{slp} + \frac{E^h_{loss}}{1h}, \tag{3.6}$$

where $\bar{N}_{act}$, $\bar{N}_{idle}$ and $\bar{N}_{slp}$ are the average number of active, idle and sleeping servers, P_{max}, P_{idle} and P_{slp} are the maximal, idle and sleep power of servers. $\bar{N}_{act} + \bar{N}_{idle} + \bar{N}_{slp} = N$. In our work, we select $\bar{N}_{idle}/\bar{N}_{slp} = 1$ to provide every idle server a sleeping server for backup.

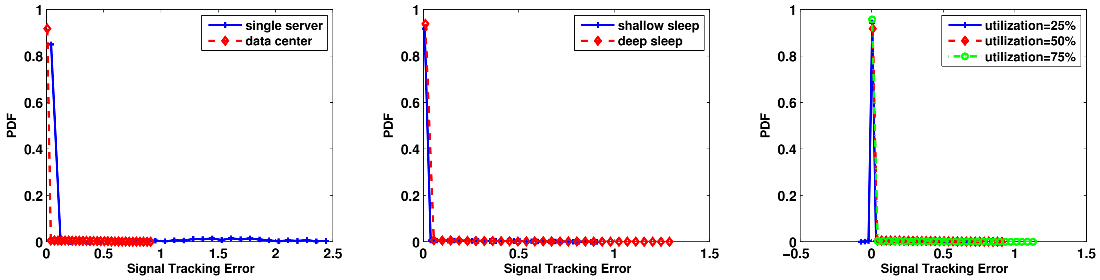
Next, we estimate reserve R. The constraints on R are:

$$\begin{aligned} \bar{P} - Ry(t) &\geq N \cdot P_{slp}, \\ \bar{P} + Ry(t) &\leq N \cdot P_{max}, \ \forall t. \end{aligned} \tag{3.7}$$

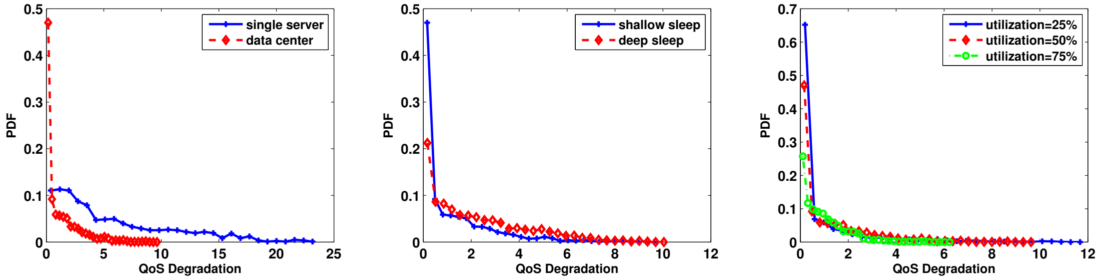
Since $y(t) \in [-1, 1]$, then:

$$R \leq min\big\{N \cdot P_{max} - \bar{P}, \ \bar{P} - N \cdot P_{slp}\big\}. \tag{3.8}$$

Prior results on single server RSR provision have shown that the value of R does not notably affect the tracking performance or the QoS degradation. Moreover, the



(a) Single Server vs. Data Center (b) Shallow vs. Deep Sleep (c) Multiple Utilization



(d) Single Server vs. Data Center (e) Shallow vs. Deep Sleep (f) Multiple Utilization

Figure 3·7: The probability density function (PDF) of tracking error (a, b, c) and job QoS degradation (d, e, f) in different cases with the best tracking policy. All the cases are with the (homogeneous) workload set of Blackscholes.

results have also shown that the optimal R is close to $min\{\bar{P}-P_{idle},\ P_{max}-\bar{P}\}$, i.e., its maximal possible value. Considering that data centers have even more flexibilities in providing RSR compared to a single server, we estimate the optimal R as the value at the bound of Eq. (3.8).

3.4.3 Experimental Results

In this section, we evaluate data center RSR provision in different scenarios with the best tracking policy. We run simulations with a data center cluster containing $N = 100$ servers. By default the data center utilization is 50%. We use the shallow sleep as the default sleep state. We simulate a 1-hour period experiment 10 times and evaluate the RSR signal tracking, workload QoS, and the energy monetary cost.

Single Server vs. Data Center

First, we compare the results of data center RSR provision with default settings to the single server results presented in Section 3.3. Figure 3·7(a) is the probability density function (PDF) of the signal tracking error $\epsilon(t)$ over time t. It shows that in most of the time, the tracking errors are close to 0 for the data center RSR provision, while for the single server the tracking errors are mostly around 0.1, i.e., 10% of the reserve R. Moreover, the data center has smaller deviation in the tracking error. The maximal tracking error of the data center is less than 1, while that of the single server reaches up to 2.5. Since sometimes ISOs put strict limitations on the peak tracking error, a data center has this additional advantage in providing RSR compared to the single server in this scenario. Overall, the data center can perform much better than a single server in RSR signal tracking. Figure 3·7(d) shows the PDF of job QoS degradation introduced in Section 3.2.3. Results demonstrate that the QoS degradation of the data center in RSR provision is on average much smaller than that of a single server.

We then check the energy cost savings from RSR provision in both cases. As introduced in Eq. (3.4), the net cost of the energy for providing RSR is $\Pi^E \bar{P} - \Pi^R R - c\Pi^R \cdot \bar{\epsilon}$. After the calculation, the energy cost saving of the optimal RSR provision compared to the case of no reserve provision (i.e., $R = 0$), for the single server is 29.7%, while for the data center is 56.8%, which is almost doubled.

Overall, providing RSR brings dramatic energy monetary savings (56.8%) to data centers, with close to zero power tracking error for most of the time, and no major QoS degradation. Compared with the single server RSR provision, the signal tracking performance, the workload QoS, and the energy cost savings are all significantly improved in the data center scenario. These results are expected, as data centers contain more flexibilities in their power and workload management that can be leveraged for RSR provision.

Shallow Sleep vs. Deep Sleep

Next, we study the impact of different server sleep states to results. Figure 3·7(b) shows the PDF of RSR signal tracking errors in the cases of the data center servers with the shallow sleep and the deep sleep, respectively. Parameters of these sleep states are introduced in Section 3.2.1. Results show that the signal tracking performance is similar and accurate in both cases. This is because the best tracking policy gives signal tracking the highest priority, and the data center power keeps tracking the signal cap no matter what types of sleep states are used for servers. Thus, the RSR signal tracking performance is not sensitive to different server sleep states.

Figure 3·7(e) is the PDF of job QoS degradation in two cases. Results show that QoS degradation in both cases is small on average. Using the shallow sleep state for servers results in smaller QoS degradation, i.e., better QoS performance, compared to using the deep sleep state. This is because the larger time delay of resuming a server with the deep sleep state leads to larger negative effects on the job servicing performance.

For a data center with the shallow sleep state, the energy monetary saving of the optimal RSR provision compared to no reserve provision is 56.8%, while with the deep sleep state, the saving is only 36.9%, for the reason that the shallow sleep state is able to react more rapidly to ISO requests than the deep sleep state, due to smaller transition delays, and thus is capable of providing more reserves without violating the signal tracking constraint. Overall, RSR provision in both cases bring significant energy monetary savings to the data center, with close to zero signal tracking error for most of the time, and small QoS degradation, while using the shallow sleep state further increases the savings and workload QoS.

Impact of Data Center Utilization

Different data centers, or data centers at different time periods may have varying utilization. We evaluate the impact of different data center utilization on results of RSR provision. Figure 3·7(c) shows the PDF of RSR signal tracking error under three different utilization settings: 25%, 50% and 75%. The figure shows that tracking performance in different utilization is similar, and most of errors are close to zero. This is because in all cases the best tracking policy gives the highest priority to signal tracking. The utilization does not have much influence on the tracking performance. Figure 3·7(f) shows the PDF of job QoS degradation under various utilization, i.e., 25%, 50% and 75%. In all three cases, the degradation is small.

We then compare the energy cost monetary savings of three utilization cases. For U=25%, 50%, 75%, the savings (the cost of the optimal RSR provision compared to that of the no reserve provision) are 78.0%, 56.8%, and 21.8%, respectively, which demonstrates that the savings decrease when the utilization increases. This is due to the reason that higher utilization requires higher average power consumption $\bar{P}$ for job servicing, which limits the regulation room for providing reserve R. However, even with 75% utilization, there is still 21.8% energy cost saving by providing RSR, which indicates significant profits of data centers in RSR participation.

Impact of Different Workloads

All previous experiments are conducted by using homogeneous Blackscholes jobs. In this part we study the data center RSR problem with different types of workloads. Table 3.4 shows the experimental results on four different workloads. We list their signal tracking statistics, QoS degradation statistics, and monetary savings. $\bar{D}_j$ and σ_{D_j} are the mean and standard deviation of QoS degradation for workload type j, $\bar{\epsilon}$ and σ_ϵ are the mean and standard deviation of the tracking error. The results show

Table 3.4: Performance and savings of data center RSR provision with different types of workloads using the best tracking policy.

	Blackscholes	Canneal	Streamcluster	Facesim
$\bar{P}/kW$	9.75	9.71	9.84	9.84
R / kW	5.54	4.98	5.46	5.11
$\bar{D}_j$	1.13	1.13	0.21	0.22
σ_{D_j}	1.54	0.69	0.26	0.27
$\bar{\epsilon}$	0.03	0.03	0.03	0.03
σ_ϵ	0.10	0.09	0.09	0.09
$R/\bar{P}$	56.8%	51.3%	55.5%	52.0%

that the signal tracking performance is not influenced by the workload type, while the QoS degradation is. From the table, workloads with longer shortest possible processing time, i.e., $T_{min,j}$, such as Streamcluster and Facesim (whose $T_{min,j}$ are larger than 100 seconds, while $T_{min,j}$ of Blackscholes and Canneal are only 20-40 seconds), have smaller QoS performance degradation. This is because the waiting time is relatively short (compared to the processing time) for workloads with larger $T_{min,j}$. Since our policy applies rules (e.g., P_{min}) to guarantee the job processing time, waiting time becomes the major uncertainty in QoS degradation. Overall, both the QoS degradation and the tracking error with all types of workloads are small. In addition, in all cases, data centers can achieve approximately 50% monetary savings (based on the $R/\bar{P}$ rate from the table). Hence data center level RSR is expected to have small tracking errors and QoS degradation along with dramatic monetary savings for a broad range of workloads.

3.5 Comparison of Energy Cost Saving Strategies

To better evaluate the energy cost savings from RSR provision, in this section, we compare the energy consumption, peak power and energy monetary costs (i.e., monthly electricity bill) of a number of advanced data center energy cost saving strategies, including multiple power management policies, and data center participation in a variety of DR programs. Since the main purpose is to investigate the capabilities on energy cost savings, in this section we apply loose QoS constraints on these strategies, to offer more flexibility for them to achieve cost savings. For simplicity, we consider the scenario with the homogeneous workload. The strategies that we study include:

- *All-on*: The data center does not participate in DR, and servers in data center are always turned on and never put to sleep, no matter the workload situation. This is one of the typical policies implemented in today's data centers, in order to guarantee the best workload QoS. However, large amount of energy is wasted.
- *SoftReactive*: The data center does not participate in DR, but servers are smartly put into sleep state to save energy, if they have been idle longer than a timeout threshold. This policy is introduced in Gandhi et al.'s work (Gandhi et al., 2012).
- *QoS-feedback*: *SoftReactive* does not take the job QoS into account while making decisions in server state transition. It simply wakes up equal number of servers to the number of arrival jobs at every time interval. If large QoS degradation is tolerable, then more energy could be potentially saved with a better policy. We introduce a *QoS-feedback* policy that is based on the *SoftReactive*, but applies the real-time workload QoS as feedback in decision making. The main idea of *QoS-feedback* is to determine the minimal number of active servers needed at time t, based on the current length of job queue and the overall QoS performance till t. The detail of the policy is referred to prior work (Chen et al., 2014b).
- *PeakShaving*: Participating the peak shaving program helps the data center eliminate the peak power so as to reduce the costs. We study the savings from data center peak shaving with a *PeakShaving* policy that leverages both server power

capping and server provisioning. Assuming the original peak power of the data center is P_{peak}, and a β percent of peak is required to be shaved to, i.e., during the peak shaving time period (i.e., an hour or a month), the data center has a strict power cap, βP_{peak} that is not allowed to be violated. Unlike the ISO signal power constraint in RSR that is dynamically changed, the power constraint in peak shaving program is fixed at βP_{peak} during the time period. Moreover, in RSR, data centers track the power signal with some degrees of tolerable tracking error, while in peak shaving, though the power consumption is strictly capped at βP_{peak}, there is no further constraint on power consumption as long as the power is lower than the cap. The *PeakShaving* policy is slightly modified from the best tracking policy for RSR provision introduced in Section 3.4. The detail of the *PeakShaving* policy is referred to prior work (Chen et al., 2014b).

- *RS*: The data center participates in RSR provision with the best tracking policy introduced in Section 3.4.

- *FC*: The data center participates in the frequency control (FC) introduced in Section 2.2. Today, FC is provided by generators through annual contracts, and there is no short term market price discovery for it. However, in anticipation of markets evolving in this direction, we assume for purposes of studying data center FC participation. In contrast to RSR provision that uses a centralized ISO signal broadcast every few seconds, the signal of FC is generated based on the local frequency deviation observation, and typically varies continuously, or changes much faster (e.g., 10x) than the ISO signal in RSR. In addition, demand side in FC is required to react immediately and exactly to follow the dynamics of the signal with its maximal possible capability. These two requirements cause much more difficulties for demand side to participate in FC than RSR. However, the price of reserves in FC is in anticipation much higher (e.g., 5x) than that of reserves in RSR, which could lead to more energy cost savings for demand side. In our *FC* policy, the data center only leverages server dynamic power management techniques (i.e., CPU resource limits), and does not apply server provisioning as the control knob, for the reason that the delay of resuming a server from the sleep state is too large to meet the requirements of FC. Therefore, only active servers can provide FC reserves. Today, DVFS can be modulated with μs-level overhead, and CPU resource limits can be modulated at ms-level in current hypervisors (Gong et al., 2010), thus, practically at real-time for our purposes. We expect future hypervisors or OS to provide finer granularity, lower overhead resource control options. The main idea of the *FC* policy is, given the workload information and data center utilization U, we estimate the number of servers that are needed to be activated during the hour. These servers are always turned on and never put to sleep. We put all the rest servers into sleep state and do not use them during the hour. The detail of the policy is referred to prior work (Chen et al., 2014b).

We conduct experiments with a 1000-server data center to compare above listed strategies. Two types of server sleep modes are used in comparison: the shallow sleep (the default setting) and the deep sleep, as introduced in Section 3.2.1. By default the data center is with 50% utilization, and the workload trace is generated from a homogeneous set of the Streamcluster application. All policies are under the same SLA constraint $(Q_j, \eta_j) = (2, 95\%)$. The price of the energy that we use in the comparison is $\Pi^E = 10.7$cent/kWh, and the peak power price is Π^P=12 \$/kW (monthly) based on prior work (Govindan et al., 2011). For emerging smart grid programs, e.g., RSR and FC, peak power is not charged separately. In order to make a fair comparison, we calculate the converted clearing price of energy in RSR and FC, i.e., Π^E_{cvt}, by taking peak power price into account, as follows:

$$\Pi^E_{cvt} = \frac{\Pi^E \cdot E_m + \Pi^P \cdot P^{peak}_m}{E_m} \tag{3.9}$$

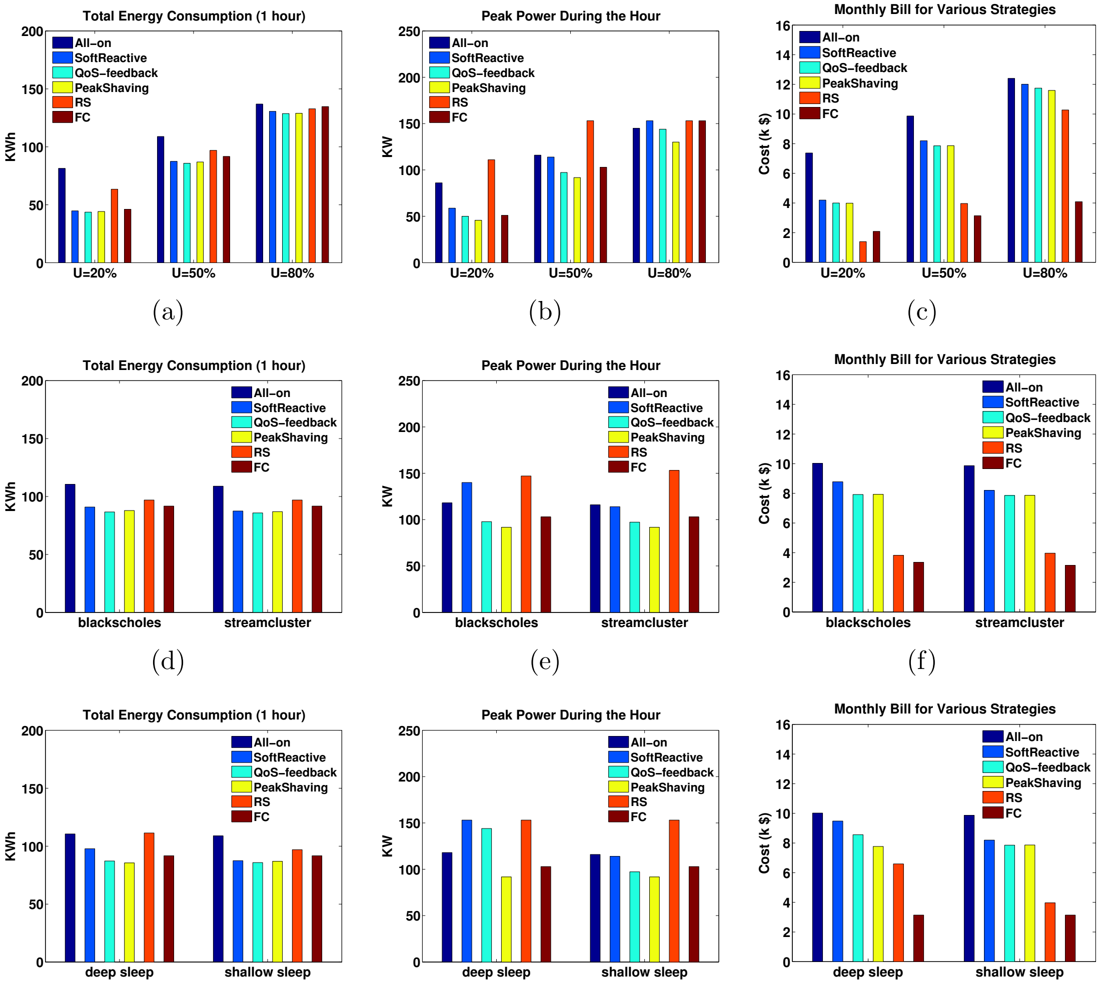
where E_m is the monthly energy consumption and P^{peak}_m is the peak power in the month. In our experiment, we assume that a monthly power trace is $24 \cdot 30$ repetitions of an hourly power trace, then we have $E_m = 24 \cdot 30 \cdot E_h$ and $P^{peak}_m = P^{peak}_h$, where E_h and P^{peak}_h are energy consumption and the peak power of the hourly power trace. After the conversion[4], Π^E_{cvt}=12.58 cent/kWh. In addition, we assume that the prices of reserves are: $\Pi^{R_2} = \Pi^E_{cvt}$ in RSR, and $\Pi^{R_1} = 5\Pi^E_{cvt}$ in FC.

The policies evaluated in the experiment include: *All-on, SoftReactive, QoS-feedback, PeakShaving, RS* and *FC*. For *All-on, SoftReactive, QoS-feedback* and *PeakShaving*, the total data center monthly electricity bill is calculated as the sum of the cost on monthly energy use and the cost on peak power, as $\Pi^E \cdot E_m + \Pi^P \cdot P^{peak}_m$, while for *RS* and *FC*, the electricity bill is calculated as $24 \cdot 30 \cdot (\Pi^E_{cvt}\bar{P}_2 - \Pi^{R_2}R_2 + \Pi^{\epsilon} \cdot \bar{\epsilon})$, and $24 \cdot 30 \cdot (\Pi^E_{cvt}\bar{P}_1 - \Pi^{R_1}R_1)$, where $\bar{P}_2$, $\bar{P}_1$, R_2, R_1 are the average power consumption and reserves in the bid, for *RS* and *FC* respectively. Figure 3·8(a) to 3·8(i) are results of data center hourly energy consumption, peak power, and monthly bill for listed energy saving strategies in different scenarios. For the energy consumption, we also calculate the *Oracle* as a baseline in comparison. *Oracle* is the minimal energy required for the data center to finish all job servicing, which maximizes the number of servers in the sleep state, assuming all the job arrivals are known ahead.

Figure 3·8(a) shows the results of data center hourly energy consumption for listed strategies under different data center utilization (i.e., 20%, 50% and 80%).

---

[4] The hourly power trace used to do the price conversion in Eq. (3.9) is generated in the following way: a workload arrival trace is first randomly generated, in the scenario of 50% data center utilization, with the homogeneous Streamcluster workload. Then the workload trace is served with the *All-on* policy.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 3·8: Comparison of hourly energy consumption, peak power and monthly electricity bill of different energy cost reduction strategies in various scenarios. (a), (b), and (c) are results under different data center utilization U = 20%, 50% and 80%; (d), (e), and (f) are results with different workload types, i.e., Blackscholes and Streamcluster; (g), (h), and (i) are results with servers using shallow or deep sleep states.

From the figure, *QoS-feedback* always achieves the lowest energy consumption. It saves 8% - 46.4% energy costs comparing to *All-on*, and is only 1.2% - 2.4% greater than the *Oracle*. The energy consumption of *SoftReactive* is close to *QoS-feedback* and *Oracle*. It is interesting to see that *PeakShaving* also has relatively low energy consumption. When increasing the data center utilization, the differences in energy consumption of various strategies get smaller, because the flexibilities in data center energy consumption with high utilization are small.

Figure 3·8(b) shows that *PeakShaving* always achieves the lowest peak power in all scenarios, which is as expected. It reduces the peak power around 10.3% -46.8% comparing to *All-on*. *RS* always achieves the highest peak power. This is because in order to maximize the reserve provision and thus maximize the monetary savings, *RS* tends to achieve a very large dynamic power range. For *FC*, however, since we put spare servers always in the sleep state and only regulate those active servers, the peak power is not as high as that of *RS*.

Figure 3·8(c) shows the data center monthly bill of different strategies. Results present that comparing to *All-on*, all the other strategies have smaller bills. Among them *RS* and *FC* save the most. *RS* saves from 17.1% to 81.2%, and *FC* saves from 67.1% to 71.7%. When utilization is low, *RS* saves the most, and when utilization is getting higher, *FC* starts to outperform. This is because *RS* leverages different server power states to provide reserves, and when utilization is lower, there is more flexible room for *RS* to enlarge reserves and savings. However, *FC* does not utilize server power states in reserve provision, hence its savings are not that sensitive to utilization.

Figure 3·8(d) to 3·8(f) compare the hourly energy consumption, peak power and monthly bill of listed strategies with two different types of workloads, i.e., Blackscholes and Streamcluster, at 50% utilization. We select these two types of workloads to compare because they have quite different profiles in terms of the power-throughput curve, processing time, etc. These figures show that there are no notable differences in all the results between these two workloads. Thus, the results of these strategies are not sensitive to types of workloads.

Figure 3·8(g) to 3·8(i) are results of using different sleep states: the shallow sleep and the deep sleep. One notable change is, the monthly bill (in Figure 3·8(i)) of *RS* increases by 66.4% from using shallow sleep to using deep sleep, demonstrating that the shallow sleep state is more efficient for RSR provision in bill reduction. The bill of *FC* is not sensitive to different sleep states, as it does not utilize these states in reserve provision. In addition, there are notable increases in peak power of *SoftReactive* and *QoS-feedback* when the deep sleep state is used. This is because that resuming servers from the deep sleep state takes longer time (e.g., 200 sec), during which servers are at the maximal power. Thus, servers have higher probabilities of staying in a constant high power state, leading to higher peak power of the data center.

Overall, among all strategies, participating in emerging smart grid DR programs, such as *RS* and *FC*, helps data centers achieve the largest electricity bill savings. When the utilization of data center is high or the deep sleep state is applied, *FC* outperforms *RS*. Otherwise, *RS* offers the largest savings. Savings of all strategies are insensitive to types of workloads.

3.6 The Stochastic Dynamic Programming (DP) Runtime Policy

The best tracking policy does not explicitly consider workload QoS constraints, and thus is easy to fail to guarantee workload servicing performance. In fact, workload QoS is one of the key metrics in evaluation of today's data center and cloud servicing. In this section, we introduce a *stochastic dynamic programming (DP)* runtime policy that optimizes the trade off between the signal tracking and the workload QoS. For simplicity, we consider the homogeneous workload, and mainly focus on two most common states of the server in emerging data centers – the active state and the idle state, while leaving rest of states to be considered in future work.

3.6.1 The Formulation of Stochastic DP Problem

Since we assume the homogeneous workload, we assume that all servers in active state have the same controllable service rate $u(t)$ at time t, i.e., the total data center power budget is always uniformly distributed to each server, so that the fairness among all servers is kept. The period cost function of the stochastic DP is composed of (i) the cost of inaccurate RSR signal tracking, i.e., $PC_{track}(t)$, characterized by the deviation between the data center power consumption $P_{con}(t)$ and the targeted power $P_{tgt}(t)$ based on the RSR signal $y(t)$, and (ii) the cost on QoS degradation, i.e., $PC_{QoSD}(t)$, characterized by the PDF of QoS degradation. In later discussion, we introduce to represent the PDF of QoS degradation by its mean and variance.

(i) Tracking cost $PC_{track}(t)$: denoting the service rate of an individual server by $u_s(t)$, it has been shown that the server power consumption $P_s(t)$ is linearly related to $u_s(t)$ with a function $f_p(\cdot)$ in Eq. (3.2). Since all servers operate at the same controllable service rate $u(t)$ for fairness, the whole data center energy consumption is $Nf_p(u(t))$, where N represents the total number of servers in data center. Given the RSR signal $y(t)$, the tracking error period cost is defined as:

$$PC_{track}(t) = \Pi^{\epsilon}|Nf_p(u(t)) - (\bar{P} + y(t)R)|, \tag{3.10}$$

where Π^{ϵ} is a constant, representing the penalty price on per unit of tracking error.

(ii) Cost on QoS degradation $PC_{QoSD}(t)$: we start with simulating a data center with $N = 1000$ servers extensively to characterize the distribution of the dynamic QoS degradation for each 4 seconds[5]. Assuming each server's maximal possible service rate is $u_{\max}$, we simulate the scenario that jobs arrive following a Poisson process with the parameter $\lambda = 50\% \cdot N u_{\max}$, where $N u_{\max}$ represents the maximal processing capability of the data center. A job arrival rate at 50% of maximal capability is selected here for the reason that an utilization around 50% is a typical scenario in emerging data centers.

Based on queuing theory, in order to guarantee that the system is stable, the average service rate of the whole data center, i.e., $N\bar{u}$, should be greater than the job arrival rate $\lambda = N \cdot 0.5 u_{\max}$. Hence the constraints on the data center RSR bidding values $(\bar{P}, R)$, where $\bar{P} = N f_p(\bar{u})$, are as follows:

$$\begin{aligned} & N f_p(u_{\max}) \geq \bar{P} > N f_p(0.5 u_{\max}), \\ & \min\left\{N f_p(u_{\max}) - \bar{P}, \bar{P} - N P_{idle}\right\} \geq R \geq 0. \end{aligned} \tag{3.11}$$

We simulate by using a 24-hour historical PJM RSR signal data (PJM, 2013) as $y(t)$, and test on different selections of $(\bar{P}, R)$ that satisfy Eq. (3.11). In addition, since different control policies lead to varying tracking errors, for the general purpose we involve the tracking error $\epsilon(t) = N f_p(u(t)) - (\bar{P} + y(t)R)$ as a Gaussian random variable in simulation, i.e., $\epsilon(t) \sim N\big(\mu_\epsilon(t), \sigma_\epsilon^2(t)\big)$, where $\mu_\epsilon(t)$ is changed for every $t = 5$ minutes, obeying a uniform distribution as $\mu_\epsilon(t) \sim U(-0.1R, 0.1R)$, and $\sigma_\epsilon^2(t) = |4\mu_\epsilon(t)|$. During the simulation, jobs are served with service rate $u(t)$ that is calculated based on the assigned data center power budget $\bar{P} + y(t)R + \epsilon(t)$. We record the QoS degradation of every job that departs the system in the 4-second interval to generate the distribution of QoS degradation for every 4 seconds.

Simulation results show that in every 4 seconds the QoS degradation is uniformly distributed. Therefore it is necessary and sufficient to characterize the PDF by its

---

[5] As mentioned in Section 2.2, 4 seconds is the frequency of the RSR signal regulated. For the rest of this section, by default the time interval t is every 4 seconds.

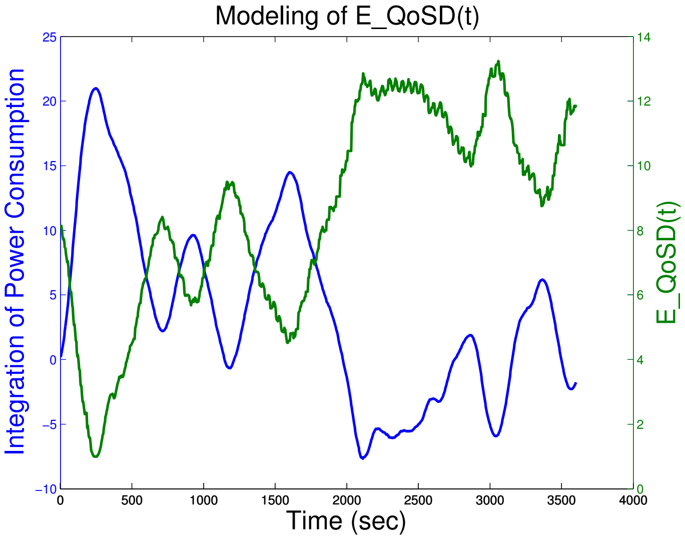


Figure 3·9: Mean of the QoS degradation is characterized by the integration of the past power consumption. Strong anti-correlation -0.97 is found between two curves.

mean and variance. We begin by formulating the mean of QoS degradation $\text{E\_QoSD}(t)$ for every 4 seconds[6]. Based on the standard queuing theory, the mean of QoS degradation depends only on the mean of queuing length μ_W of the system, when the data center does not provide RSR and consumes power steadily at level $\bar{P}$. When providing RSR, if the power consumption $Nf_p(u(t))$ is higher than $\bar{P}$, then it results in a smaller value of μ_W and therefore smaller $\text{E\_QoSD}(t)$, and vice versa. We further observe a strong anti-correlation (-0.97) between the integration of the past history of power consumption (with $\bar{P}$ as the reference value), i.e., $\int\limits_0^t (Nf_p(u(\tau)) - \bar{P})d\tau$ and $\text{E\_QoSD}(t)$, which is shown in Figure 3·9. Hence, we propose to model $\text{E\_QoSD}(t)$ with linear regression as follows:

$$\text{E\_QoSD}(t) = \alpha_{DP} \int\limits_0^t (Nf_p(u(\tau)) - \bar{P})d\tau + g(\mu_W) + \omega_{DP,1}, \tag{3.12}$$

---

[6] $\text{E\_QoSD}(t)$ denotes the mean of the QoS degradation, i.e., D_i, of all jobs i that depart during the 4-second interval at time t.

where $g(\mu_W)$ is the proper function that transforms the mean of queue length to mean of system degradation, α_{DP} can be determined from simulation, and $\omega_{DP,1}$ is a zero mean random variable with known variance. Since $f_p(u(t)) = k_j u(t) + P_{static}$ from Eq. (3.2), and $\bar{P} = N f_p(\bar{u}) = N(k_j \bar{u} + P_{static})$, Eq. (3.12) can be simplified as:

$$\text{E\_QoSD}(t) = \alpha_{DP} N k_j \int_0^t (u(\tau) - \bar{u}) d\tau + g(\mu_W) + \omega_{DP,1}, \tag{3.13}$$

in which we linearly transform the integration of power consumption to the integration of the service rate.

The variance of the QoS degradation, V_QoSD(t), is affected by the number of job departures Dep(t) in every 4 seconds from the observation[7]. A larger Dep(t) results in more sample uncertainties, and therefore larger V_QoSD(t). Figure 3·10 is the scattered plot between Dep(t) and V_QoSD(t), whose correlation is 0.94. With linear regression we have:

$$\text{V\_QoSD}(t) = \beta_{DP}(\text{Dep}(t) - 1) + \omega_{DP,2}, \tag{3.14}$$

where Dep(t) can be estimated as a Poisson random variable with $\lambda = N u(t) \Delta_t$ ($\Delta_t = 4$ seconds) based on simulation results. β_{DP} and $\omega_{DP,2}$ can be determined from simulation.

Given E_QoSD(t) and V_QoSD(t), the PDF of the uniformly distributed QoSD(t) is

$$p(\text{QoSD}(t)) = \begin{cases} \frac{1}{\sqrt{12\text{V\_QoSD}(t)}} & \text{QoSD}(t) \in [a, b] \\ 0 & \text{otherwise} \end{cases}, \tag{3.15}$$

---

[7] V_QoSD(t) denotes the variance of the QoS degradation, i.e., D_i, of all jobs i that depart during the 4 second interval at time t.

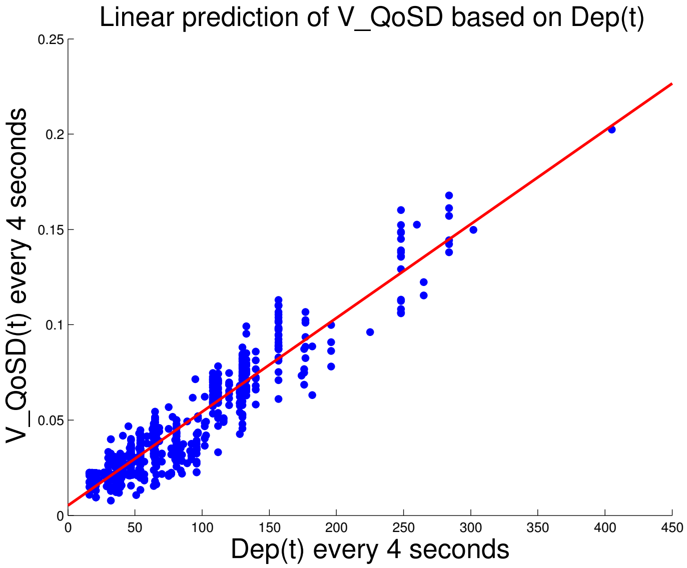


Figure 3·10: Variance of QoS degradation is characterized by the number of job departure (i.e., finished) in 4 seconds. The higher departure number results in higher uncertainty to the system, and thus higher variance.

where the lower and upper bounds are

$$\begin{aligned} a &= \text{E\_QoSD}(t) - \sqrt{3\text{V\_QoSD}(t)}, \\ b &= \text{E\_QoSD}(t) + \sqrt{3\text{V\_QoSD}(t)}. \end{aligned} \tag{3.16}$$

If the data center operator signs a contract with users in which a penalty $C(\text{QoSD}(t))$ is added when the QoS degradation exceeds a pre-defined level Q, then the expected period cost per job departure incurred by $\text{QoSD}(t)$ is

$$\int_{Q}^{\infty} p(\text{QoSD}(t))C(\text{QoSD}(t))d\text{QoSD}(t) - \Pi^{SV}, \tag{3.17}$$

where Π^{SV} represents the credit earned from per job departure. The overall period cost of QoS degradation for every 4 seconds equals to the expected cost of all job departures in that 4-second interval:

$$\begin{aligned}&PC_{QoSD}(t) = \\ &\mathbf{E}\Big[\text{Dep}(t)\Big(\int_{Q}^{\infty} p(\text{QoSD}(t))C(\text{QoSD}(t))d\text{QoSD}(t) - \Pi^{SV}\Big)\Big].\end{aligned} \tag{3.18}$$

Finally, based on Eq. (3.10) and Eq. (3.18), the total period cost function for every 4 seconds is

$$\begin{aligned}&PC_{total}(t) = PC_{track}(t) + PC_{QoSD}(t) = \Pi^{\epsilon}|Nf_p(u(t)) - (\bar{P} + y(t)R)| + \\ &\mathbf{E}\Big[\text{Dep}(t)\Big(\int_{Q}^{\infty} p(\text{QoSD}(t))C(\text{QoSD}(t))d\text{QoSD}(t) - \Pi^{SV}\Big)\Big].\end{aligned} \tag{3.19}$$

Next, we formulate the state dynamics of the stochastic DP. Clearly E_QoSD(t) in Eq. (3.13) is not Markov with respect to $u(t)$. We transform the variable into a memoryless one by adding an auxiliary variable $z(t)$ representing the integration of the service rate $u(t)$ corresponding to $\bar{u}$ up to time t. Letting $z(0) = 0$, we have the dynamic of $z(t)$ as

$$z(t+4) = z(t) + (u(t) - \bar{u}). \tag{3.20}$$

Substituting $z(t)$ into Eq. (3.13), we have

$$\text{E\_QoSD}(t) = \alpha_{DP} N k_j z(t) + g(\mu_W) + \omega_{DP,1}. \tag{3.21}$$

The dynamics of the RSR signal can be formulated by a Markov chain with two states: the value of $y(t)$ and the sign of $y(t) - y(t-4)$, namely $d(t)$, representing the direction of the signal changes at time t. Conceptually this can be represented as:

$$\begin{aligned} y(t+4) &= f_1(y(t), d(t)), \\ d(t+4) &= f_2(y(t), d(t)). \end{aligned} \tag{3.22}$$

Since the statistical behavior of the RSR signal is known ahead, function f_1 and

f_2 can be calculated in advance. They can also be mined from historical ISO RSR signal data. Detailed discussion on Eq. (3.22) is referred to the prior work (Zhang et al., 2014a).

We formulate the stochastic DP problem as a discounted cost infinite horizon DP. If we denote the value function as $J(y, d, z)$ and the overall state dynamics by

$$x(t+4) = f(x(t)), \tag{3.23}$$

where $x(t)$ is composed of $\{y(t), d(t), z(t)\}$, then the Bellman Equation is

$$J(y, d, z) = PC_{total}(y, d, z, u) + \eta_{DP}\mathbf{E}\Big[J(f(y, d, z, u))\Big], \tag{3.24}$$

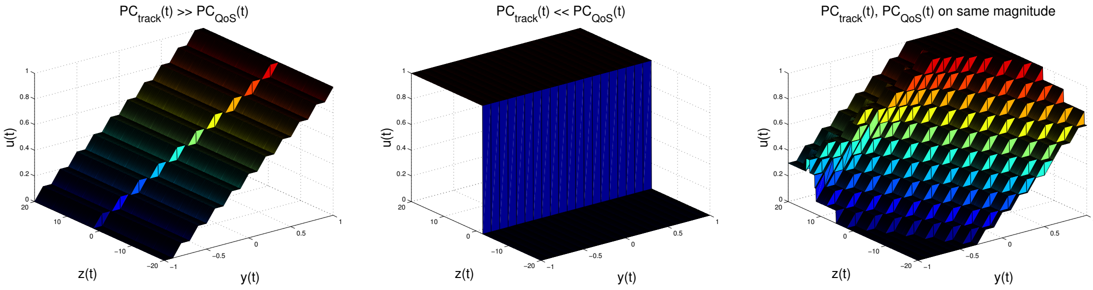
where η_{DP} is the discounted rate.

To conclude, the state variables are $\{y(t), d(t), z(t)\}$, the control variable is $u(t)$, the disturbances are E_QoSD(t), V_QoSD(t), and the discounted cost infinite horizon DP is to solve the following problem

$$\begin{aligned} \min \quad & Eq.\ (3.24)\ \text{ over }\ u \\ \text{s.t.} \quad & Eq.\ (3.14), (3.15), (3.20), (3.21), (3.22). \end{aligned} \tag{3.25}$$

3.6.2 Policy Details

We then use the simulation method (i.e., the *value iteration* method) to solve for the optimal control policy u of our discounted rate infinite horizon DP problem. We normalize and discretize the control variable $u(t)$ into 11 levels in the range of [0,1], with the granularity at 0.1. For the state variable $z(t)$ introduced in Eq. (3.20), we simulate extensively and study its distribution to acquire its possible range. Based on the distribution, a range of [-20, 20] can include more than 95% values of $z(t)$. For $|z(t)| > 20$, we truncate them to 20. Since $z(t)$ is the integral of $u(t)$, we discretize $z(t)$ using the same granularity as $u(t)$. We discretize the state variable $y(t)$ at the



(a) (b) (c)

Figure 3·11: The optimal stochastic DP policies $u(t)$ via $y(t)$ and $z(t)$ given $d(t) = 1$, of three cases: (a) $PC_{track}(t) >> PC_{QoSD}(t)$, i.e., the tracking cost dominates in the overall period cost function; (b) $PC_{track}(t) << PC_{QoSD}(t)$, i.e., the cost of QoS degradation dominates in the overall period cost function; (c) $PC_{track}(t)$ and $PC_{QoSD}(t)$ are on the same order of magnitude.

granularity of 0.1. Since the state variable $d(t) = \pm 1$, there are 16842 ($= 401 \times 21 \times 2$) different states in total.

By understanding the real-life data center SLA, we define the cost function of the QoS degradation in Eq. (3.17) as

$$C(\text{QoSD}(t)) = \begin{cases} 0, & \text{if} \;\; \text{QoSD}(t) \in [1, Q) \\ \Pi^D \text{QoSD}(t), & \text{otherwise} \end{cases}, \tag{3.26}$$

which is a discontinuous function. Π^D is a constant, representing the penalty price on per unit of the QoS degradation. We select the threshold $Q = 3$ in simulation. Π^D here and Π^{SV} in Eq. (3.18) are estimated based on the price information of Amazon Web Service (AWS) (Amazon, 2015). In general, Π^D and Π^{SV} have the same order of magnitude.

The optimal policy can be quite different while selecting different values of Π^ϵ and Π^D. A large Π^ϵ can lead to $PC_{track}(t) >> PC_{QoSD}(t)$, while large Π^D can have $PC_{track}(t) << PC_{QoSD}(t)$. Figure 3·11 shows the optimal control policy u via key state space variables $y(t)$ and $z(t)$ given $d(t) = 1$, of the following three cases:

(i) Π^{ϵ} is large, i.e., $PC_{track}(t) >> PC_{QoSD}(t)$. In this case the cost of tracking error is much larger than that of the QoS degradation, the optimal policy tends to always track the RSR signal $y(t)$ as accurate as possible to minimize the overall costs. So the policy is sensitive to and monotonically varies with $y(t)$, and is almost independent of $z(t)$;

(ii) Π^{D} is large, i.e., $PC_{track}(t) << PC_{QoSD}(t)$. In this case the cost of tracking error is much smaller than that of the QoS degradation, the optimal feedback policy is a bang-bang controller that either sets $u(t)$ at the minimal or at the maximal level. Specifically, if the mean of QoS degradation E_QoSD(t) is large because of a small $z(t)$, then $u(t) = 0$ and the policy decreases the number of departure jobs at t, i.e., Dep(t). If E_QoSD(t) is small because of a large $z(t)$, then $u(t) = 1$ and the policy increases the number of departure jobs at t, so as to minimize the overall costs. Overall, the optimal policy in this case is only sensitive to $z(t)$ and is independent of $y(t)$;

(iii) $PC_{track}(t)$ and $PC_{QoSD}(t)$ have the same order of magnitude. While **(i)** and **(ii)** are two extreme cases, case **(iii)** requires to balance between the signal tracking costs and the QoS degradation costs. From Figure 3·11(c) we find that the optimal policy depends on both $z(t)$ and $y(t)$. The policy shows that: 1) for the same $z(t)$, the optimal service rate $u(t)$ increases when the signal $y(t)$ increases, so as to better track the signal; 2) for the same $y(t)$, the optimal service rate generally increases as $z(t)$ increases, which shows that when the mean of QoS degradation is small, the optimal policy tries to finish and depart more jobs during that moment; 3) there is a non-monotonic behavior of the optimal policy around $z(t) = 10$ to $z(t) = 15$. This region of $z(t)$ corresponds to the region of E_QoSD(t) around threshold Q in Eq. (3.26), which is the discontinuous turning point of the QoS degradation penalty cost function, while below which there is no degradation penalty and above which the penalty linearly increases. Such non-monotonic behavior of $u(t)$ can be explained as follows: When $z(t)$ corresponds to E_QoSD(t) that is near the left extreme of threshold Q, at which there is still no penalty of QoS degradation, the policy applies larger service rate $u(t)$ to finish and depart more jobs to minimize the overall costs. When $z(t)$ is larger, e.g., $z(t) = 20$, however, the data center operator does not necessarily use large service rate, as there would be also no penalty if jobs are finished and depart later when $z(t+\Delta) = 15$ for small Δ. Instead, the system can focus more on eliminating tracking errors at that moment to minimize the overall costs. This explains the phenomenon that the optimal service rate $u(t)$ can be larger for $z(t) = 15$ than $z(t) = 20$.

The optimal policy of $d(t) = -1$ is similar to that of $d(t) = 1$, except that there is a small shift in the figure along the direction of $z(t)$ axis. This is because that when $d(t) = -1$, there is a higher probability that the RSR signal $y(t)$ is going to decrease in the future than when $d(t) = 1$. In order to eliminate the overall tracking error, the optimal policy of $d(t) = -1$ prefers lower $u(t)$ than that of $d(t) = 1$. Overall, the shift is small, which shows that the policy is not very sensitive to the state variable $d(t)$.

3.6.3 Energy and Reserve Bidding

Acquiring the optimal policy, we then study the optimal hour-ahead bidding strategies for the data center operator in the energy and reserve market. For the scenario of the data center with 1000 servers and the utilization of 50%, Eq. (3.11) provides the constraints of the bidding values. Obeying the constraints, we run simulations with different $(\bar{P}, R)$ and measure the data center's hourly overall bill as

$$B(\bar{P}, R) = \Pi^E \bar{P} - \Pi^R R + J_{total}(\bar{P}, R), \tag{3.27}$$

where $J_{total}(\bar{P}, R)$ is the summed value of the tracking error cost and the cost of QoS degradation in DP formulation, introduced in Section 3.6.4.

For simplicity of notation, we denote $Nf_p(0.5u_{\max})$, i.e., the lower bound of $\bar{P}$ in Eq. (3.11) as P_{lb}. Table 3.5 shows the overall hourly bill (in dollars) of a 1000-server data center with $\bar{P} = 1.001$, 1.1 and 1.2 P_{lb} respectively in each row[8]. For each selected value of $\bar{P}$, the maximal possible reserve value is: $R_{\max} = \min(Nf_p(u_{\max}) - \bar{P}, \bar{P} - NP_{idle})$ from Eq. (3.11). In the table, we measure the bill via $R = 20\%$, 40%, 60%, 80% and 100% $R_{\max}$ respectively, for each $\bar{P}$. The price information used in simulation is estimated based on real power market (PJM, 2013; Aikema et al., 2012) and AWS (Amazon, 2015) data, i.e., $\Pi^E = \Pi^R = \Pi^\epsilon = 0.2\$/\text{kWh}$, $\Pi^D = \Pi^{SV} = 0.1\$/\text{h}$. These price values lead $PC_{track}(t)$ and $PC_{QoSD}(t)$ to share the same order of magnitude, and thus the scenario falls in the category (iii) introduced in Section 3.6.2.

The table shows that, satisfying the constraints in Eq. (3.11), the overall hourly bill of the data center in RSR provision increases monotonously as $\bar{P}$ increases, and decreases monotonously as R increases. Therefore, in order to minimize the monetary costs, the optimal bidding mechanism for the data center RSR is to choose the smallest $\bar{P}$ and the largest R that satisfy Eq. (3.11).

3.6.4 Policy Comparison

We compare the stochastic DP policy with the best tracking policy introduced in Section 3.4. We consider the same scenario for both policies: a data center with $N = 1000$ servers, jobs arrive following the Poisson distribution with the arrival rate as $\lambda = 50\% \cdot Nu_{\max}$. Since the stochastic DP policy does not use the sleep state, to make a fair comparison, we disable the server sleep state in the best tracking policy as well. We run simulation with a real 24-hour historical RSR signal data

---

[8] $1.001P_{lb}$ is selected because P_{lb} itself does not satisfy Eq. (3.11), however, a power value slightly larger than P_{lb} is able to, e.g., $1.001P_{lb}$.

Table 3.5: The hourly electricity bill of a 1000-server data center with RSR provision via different ($\bar{P}$, R), using the stochastic DP policy.

	100%	80%	60%	40%	20%
1.001	\$31.33	\$34.89	\$38.23	\$41.36	\$43.90
1.1	\$40.79	\$43.55	\$46.43	\$49.58	\$53.00
1.2	\$47.99	\$49.49	\$50.82	\$52.02	\$53.19

---

[a] 1.001, 1.1 and 1.2 represent $\bar{P} = 1.001P_{lb}$, $1.1P_{lb}$ and $1.2P_{lb}$ respectively, with $P_{lb} = Nf_p(0.5u_{\max})$.

[b] 100%, 80%, 60%, 40% and 20% represent R = $100\%R_{max}$, $80\%R_{max}$, $60\%R_{max}$, $40\%R_{max}$ and $20\%R_{max}$ respectively, with $R_{max} = \min(Nf_p(u_{\max}) - \bar{P}, \bar{P} - NP_{idle})$.

from PJM (PJM, 2013) as our $y(t)$, and use only data from the 2^{nd} to the 23^{rd} hour (data from the 1^{st} and the last hour is not clean and stable due to the effects of the initialization and termination of the experiment). We treat this 22-hour simulation as 22 repetitions of the 1-hour experiment and then measure the statistics in order to achieve statistical confidence. The price information used in simulation is the same as that introduced in Section 3.6.3.

We measure the tracking cost J_{track} and the cost of QoS degradation J_{QoSD}, and recall that the overall cost J_{total} equals to $J_{track} + J_{QoSD}$. Comparing to the best tracking policy, our DP policy incurs larger J_{track}, which is expected, as the best tracking policy tracks signal as accurate as possible. However, the stochastic DP policy leads to much smaller J_{QoSD}, due to the fact that the QoS degradation is carefully taken into account. Overall, the total cost, J_{total} of the stochastic DP policy is on average 4.55% smaller than the cost of the best tracking policy.

3.7 The EnergyQARE Runtime Policy

While the best tracking policy does not explicitly take workload QoS into account in decision making, the stochastic DP policy uses the simplified data center model and does not support multiple server power states, in this section, we introduce an

EnergyQARE, the **Energy** and **Q**oS **A**ware **R**SR **E**nabler runtime policy that handles the more general and practical scenarios for our data center model shown in Figure 3·2: the heterogeneous workloads and power budgeting, multiple server states and server provision, workload scheduling and allocation are all taken into account. In addition, the EnergyQARE policy guarantees the workload QoS by applying not only the signal tracking performance, but also the QoS as feedback. Specifically, EnergyQARE dynamically monitors signal tracking and workload QoS, and adaptively makes decisions on which of them (i.e., tracking error or QoS) should be given higher priority. Comparing with prior introduced two policies, EnergyQARE is more suitable for a general real-life data center scenario, and with tight constraints on both signal tracking and workload QoS.

3.7.1 Policy Details

The flowchart of the EnergyQARE runtime policy is shown in Figure 3·12. The principal philosophy of the policy is to dynamically make online decisions based on monitored signal tracking and workload QoS state variables. At each time t, the main state variables that are monitored include:

1. the mean of the tracking error, i.e., $\bar{\epsilon}(t)$ during a past time window $[t-T, t]$;
2. the mean of the QoS degradation, i.e., $\bar{D}_j(t)$ for each cluster j during the past time window $[t-T, t]$;
3. the total number of additional required active servers $N_{req}(t)$ to meet the SLAs based on the Little's Law (Leon-Garcial, 2008), which is highly correlated with the length of the job queue;
4. the states of jobs and servers;
5. the value of RSR signal $y(t)$.

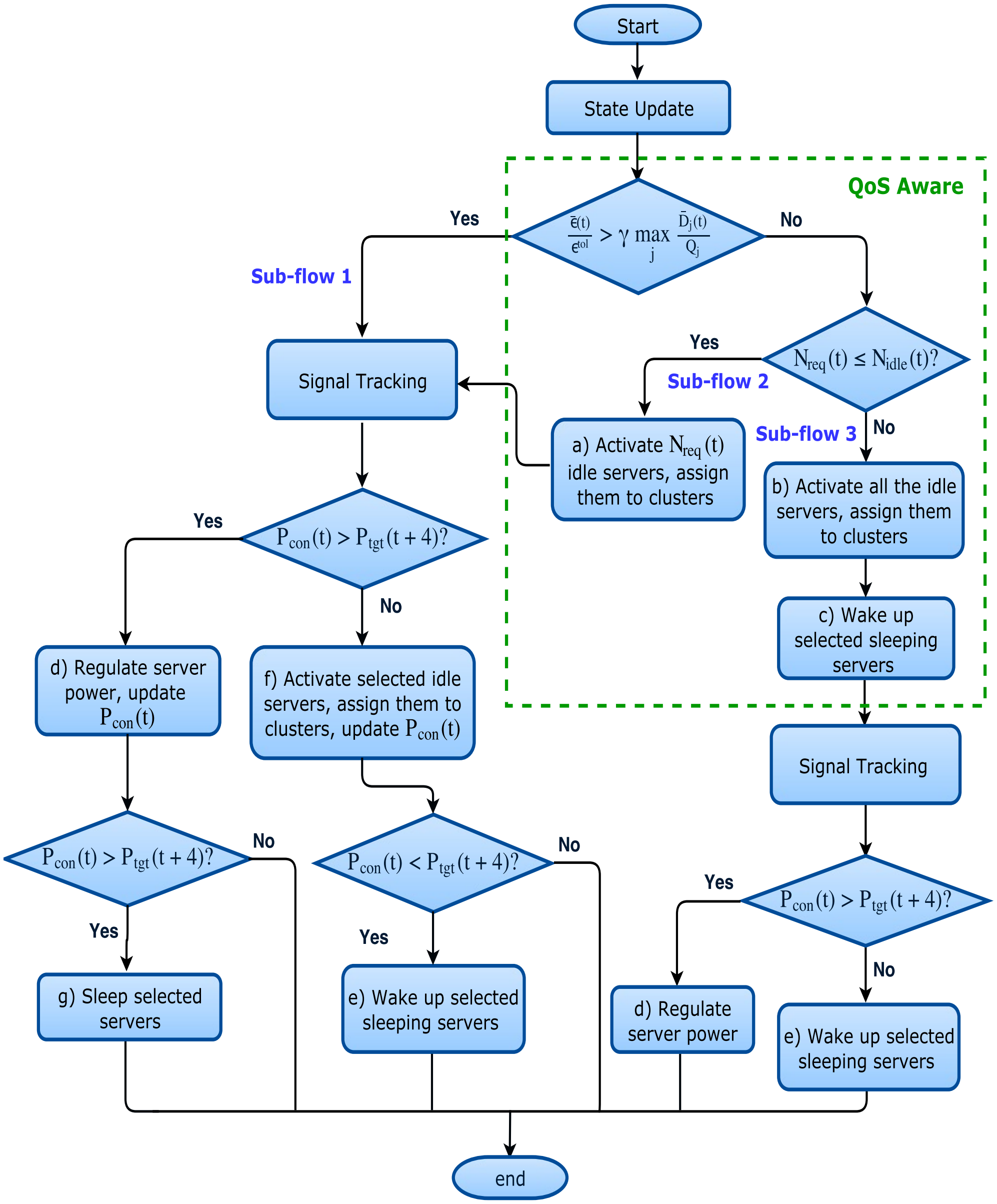

Start
State Update
QoS Aware
Yes
$\frac{\bar{\epsilon}(t)}{\epsilon^{tol}} > \gamma \max_j \frac{\bar{D}_j(t)}{Q_j}$
No
Sub-flow 1
Signal Tracking
Yes
$N_{req}(t) \leq N_{idle}(t)$?
Sub-flow 2
a) Activate $N_{req}(t)$ idle servers, assign them to clusters
Sub-flow 3
No
b) Activate all the idle servers, assign them to clusters
c) Wake up selected sleeping servers
Yes
$P_{con}(t) > P_{tgt}(t+4)$?
No
d) Regulate server power, update $P_{con}(t)$
f) Activate selected idle servers, assign them to clusters, update $P_{con}(t)$
Signal Tracking
$P_{con}(t) > P_{tgt}(t+4)$?
No
$P_{con}(t) < P_{tgt}(t+4)$?
No
Yes
Yes
Yes
$P_{con}(t) > P_{tgt}(t+4)$?
No
g) Sleep selected servers
e) Wake up selected sleeping servers
d) Regulate server power
e) Wake up selected sleeping servers
end

Figure 3·12: The flowchart of the EnergyQARE runtime policy.

Based on these state variables, EnergyQARE selects one or more from the following actions:

1. wake up sleeping servers;
2. put idle servers into sleep;
3. assign idle servers to clusters to serve waiting jobs;
4. regulate the power and service rate of the active servers.

In the following sections, we introduce the details of the state variables and actions, as well as the overall runtime policy. Table 3.6 lists the descriptions of some major symbols used in Eq. (3.28) to Eq. (3.48). Other prior defined symbols are referred to Section 3.2.

State variables in EnergyQARE

The system states are measured and updated at the beginning of each time interval t. The main state variables used for decision making in EnergyQARE are as follows:

1. The mean of the tracking error $\bar{\epsilon}(t)$. $\bar{\epsilon}(t)$ is the mean of the instantaneous signal tracking error $\epsilon(t)$ (defined in Eq. (2.1)) during a past time window $[t-T, t]$, i.e.,:

   $$\bar{\epsilon}(t) = \frac{1}{T} \sum_{\tau=t-T}^{t} \epsilon(\tau). \tag{3.28}$$

   The size of the time window T is selectable.

2. The mean of the QoS degradation $\bar{D}_j(t)$. Similar to the way that we calculate $\bar{\epsilon}(t)$, we calculate the mean of the QoS degradation $\bar{D}_j(t)$ for each cluster j at time t as an average of degradation $D_{i,j}$ introduced in Section 3.2.3 for all jobs i in cluster j that are finished their servicing and depart the system in a past time window $[t-T, t]$:

   $$\bar{D}_j(t) = \frac{\sum_{i=1}^{\mathrm{Dep}_j(t)} D_{i,j}}{\mathrm{Dep}_j(t)}, \tag{3.29}$$

Table 3.6: Descriptions of major symbols in Eq. (3.28) to Eq. (3.48).

Symbol	**Description**
$N_j(t)$	The number of servers in cluster j at t.
$N_{idle}(t)$	The number of servers in the idle pool at t.
$N_{slp}(t)$	The number of servers in the sleep pool at t.
$N_{tran}(t)$	The number of servers in the transition state at t.
$N_{req,j}(t)$	The number of additional active servers required by cluster j at t.
$N_{req}(t)$	The total number of additional active servers required for all clusters at t.
$N_{rdslp}(t)$	The number of servers that are ready to be put into sleep at t.
$N_{up}(t)$	The number of servers to be waken up at t.
$N_{toslp}(t)$	The number of servers to be put into sleep at t.
$N_{assg,j}(t)$	The number of idle servers that are assigned to cluster j at t.
$N_{assg}(t)$	The total number of idle servers that are activated and assigned to all clusters at t.
T_{res}	Time delay for waking up a server.
$T_{w,s}(t)$	Time that a server s has been in the transition state at t.
$T_{idle,s}(t)$	Time that a server s has been in the idle state at t.
T_{out}	The timeout value used to determine whether to put a server into the sleep state.
$p_j(t)$	The number of running jobs in cluster j at t.
$q_j(t)$	The number of waiting jobs in cluster j at t.
$u_j(t)$	Service rate of servers in cluster j at t.
$u_{max,j}$	The maximal possible service rate for jobs in cluster j.
$u_{min,j}(t)$	The minimal service rate for jobs in cluster j at t to meet SLAs.

where $\text{Dep}_j(t)$ is the number of jobs that are finished and depart in the time window $[t-T,t]$ in cluster j.

3. The total number of additional active servers required for all clusters $N_{req}(t)$. We first calculate the number of additional servers required by each cluster j, i.e., $N_{req,j}(t)$, for $j \in 1,2,3..,M$, where M is the number of clusters. We calculate $N_{req,j}(t)$ based on Little's Law (Leon-Garcial, 2008): we denote the number of waiting jobs and running jobs in cluster j as $q_j(t)$ and $p_j(t)$ at time t, respectively. Since each cluster contains homogeneous workload, we assume all servers in each cluster are always running at the same service rate $u_j(t)$ for cluster j, so that there is fairness of servicing among jobs of the same type.

Then the average job system time calculated from Little's Law for cluster j at time t is:

$$\bar{T}^L_{sys,j}(t) = \frac{q_j(t) + f_r(p_j(t))}{u_j(t)n_j(t)}, \tag{3.30}$$

where $n_j(t)$ is the number of servers in cluster j at time t. We use superscript L to denote that this average system time is calculated from Little's Law, so as to distinguish it with the notation $\bar{T}_{sys,j}(t)$, i.e., the average system time from the real measurement. Function $f_r(\cdot)$ is used to transform the unfinished parts of running jobs to the number of full jobs, i.e.,:

$$f_r\big(p_j(t)\big) = \sum_{i=1}^{p_j(t)} \frac{I_{r,i}(t)}{I_i}, \tag{3.31}$$

where $I_{r,i}(t)$ is the number of unfinished instructions of job i at time t, I_i is the total number of instructions of the job i. A job can be considered as a set of instructions, and finishing a job is equivalent to execute all instructions of that job.

The job system time is constrained by SLAs introduced in Eq. (3.3). We deduce a constraint on the average system time $\bar{T}^L_{sys,j}(t)$ from Eq. (3.3) as:

$$\bar{T}^L_{sys,j}(t) \leq (\beta Q_j + 1) \cdot T_{min,j}, \tag{3.32}$$

where β is the coefficient determined by the probability distribution of $T_{sys,i}$, i.e., the system time of the job i. Putting Eq. (3.30) into Eq. (3.32), we obtain a constraint on the number of servers in cluster j:

$$n_j(t) \geq \frac{q_j(t) + f_r(p_j(t))}{(\beta Q_j + 1) \cdot T_{min,j} u_j(t)}. \tag{3.33}$$

Prior work has shown that serving jobs at the maximal possible service rate u_{max} in general provides best energy efficiency (Gandhi et al., 2012; Chen et al., 2014b). Hence in this work we assume that by default all the jobs are served at their maximal possible rate, and we use $u_{max,j}$ to substitute $u_j(t)$ in Eq. (3.33). Since $u_{max,j} \cdot T_{min,j} = 1$ (recalling that $T_{min,j}$ is the shortest possible processing time when the job is served at the maximal possible service rate $u_{max,j}$), Eq. (3.33) is simplified to:

$$n_j(t) \geq \frac{q_j(t) + f_r(p_j(t))}{\beta Q_j + 1}. \tag{3.34}$$

The right hand side of Eq. (3.34) is the minimal number of required servers for cluster j at time t, in order to meet the system time constraint from Little's Law. Assuming currently there are $N_j(t)$ active servers in cluster j, the number of additional servers required for cluster j to meet the QoS constraint is:

$$N_{req,j}(t) = max\Big\{0, \quad \frac{q_j(t) + f_r(p_j(t))}{\beta Q_j + 1} - N_j(t)\Big\}. \tag{3.35}$$

The total number of required servers for all the clusters at time t, i.e., $N_{req}(t)$, is the summation of the number of the required servers in all clusters:

$$N_{req}(t) = \sum_{j=1}^{M} N_{req,j}(t). \tag{3.36}$$

In addition to these listed three state variables, the state of each job and server, as well as the RSR signal value $y(t)$ are directly monitored and recorded.

The Overall EnergyQARE Runtime Policy

In this section we explain the EnergyQARE runtime policy shown in the flowchart in Figure 3·12. In the figure, we mark each action with a letter. Similar actions are marked with the same letter.

At the beginning of each time interval t, the system states are updated. The updates include:

- Check whether any servers have finished their jobs. Servers that finished jobs are put back into the idle server pool. The QoS of finished jobs is calculated;
- Check whether any servers have finished their transition processes. Servers that have finished their transition processes are put into the idle server pool;
- Monitor new coming jobs in the waiting queue for each cluster.

After these updates, we calculate major state variables introduced in Section 3.7.1. Then we compare the mean of the tracking error (normalized by its tolerance ϵ^{tol} in Eq. (2.2)), i.e., $\frac{\bar{\epsilon}(t)}{\epsilon^{tol}}$, with the mean of the QoS degradation (normalized by its tolerance Q_j in Eq. (3.3)), i.e., $\gamma \max_j \frac{D_j(t)}{Q_j}$. Here $\max_j \frac{\bar{D}_j(t)}{Q_j}$ represents the worst mean of QoS degradation (normalized by the tolerance Q_j) among all clusters at time t. γ is the coefficient given to the QoS degradation in comparison with the tracking error. Different γ indicates different focuses in terms of tracking error and workload QoS, e.g., a small γ can be used if the tracking error is more costly. We also compare the number of required servers $N_{req}(t)$ with the number of idle servers $N_{idle}(t)$. We move from the main flow to one of the three sub-flows based on these two comparisons. The three sub-flows are:

1. $\frac{\bar{\epsilon}(t)}{\epsilon^{tol}} > \gamma \max_j \frac{\bar{D}_j(t)}{Q_j}$. In this scenario, the signal tracking performance is relatively worse than the QoS performance, so the policy selects actions to track the signal accurately. The real power $P_{con}(t)$ and the targeted power $P_{tgt}(t+4)$ are calculated and compared. If $P_{con}(t) > P_{tgt}(t+4)$, i.e., the real power needs to be reduced, then we first slow down some servers and reduce their power in Action d. After that if we still have the condition $P_{con}(t) > P_{tgt}(t+4)$, then we put some servers into sleep in Action g. We regulate the server power before putting servers into the sleep state in reducing the power, so as to avoid server transitions that may bring large cost in terms of time delay and energy

loss later when servers are required to be waken up. On the other hand, if $P_{con}(t) < P_{tgt}(t+4)$, i.e., the real power needs to be increased, then we first activate idle servers with waiting jobs and assign them to clusters in Action f. After that if we still have the condition $P_{con}(t) < P_{tgt}(t+4)$, we wake up some sleeping servers in Action e. Similarly here we give the server transition action (waking up servers in Action e) a lower priority due to the concern of additional energy loss and time delay.

2. $\frac{\bar{\epsilon}(t)}{\epsilon^{tol}} \leq \gamma \max\limits_{j} \frac{\bar{D}_j(t)}{Q_j}$, and $N_{req}(t) \leq N_{idle}(t)$. In this scenario, the QoS performance is relatively worse than the signal tracking performance, but the number of current idle servers is sufficient to improve QoS to meet the SLA requirements. We first activate and assign $N_{req}(t)$ idle servers to all clusters to serve waiting jobs in Action a. Then we focus on the signal tracking and move to Sub-flow 1.

3. $\frac{\bar{\epsilon}(t)}{\epsilon^{tol}} \leq \gamma \max\limits_{j} \frac{\bar{D}_j(t)}{Q_j}$ and $N_{req}(t) > N_{idle}(t)$. In this scenario, the QoS performance is relatively worse than the signal tracking performance, and the number of idle servers is not sufficient to meet the requirement. We first activate all idle servers with waiting jobs in Action b. Then we wake up additional servers so as to meet the requirement $N_{req}(t)$ in Action c. After that, we consider the signal tracking. If $P_{con}(t) > P_{tgt}(t+4)$, we slow down servers in Action d. Here we no longer consider to put servers to sleep, because prior to this action we have already been required to wake up servers in Action c. On the other hand, if $P_{con}(t) < P_{tgt}(t+4)$, since all the idle servers have been activated in Action b, the only action to increase $P_{con}(t)$ is to wake up additional sleeping servers in Action e.

The details of each action will be introduced in the next section.

Detailed Actions in EnergyQARE

All the actions in Figure 3·12 fall into four basic groups: wake up sleeping servers (Actions c and e), put idle servers into sleep (Action g), assign idle servers to clusters to serve waiting jobs (Actions a, b and f), and regulate the power and service rate of the active servers (Action d). Below we discuss each of them in detail.

1. Wake up sleeping servers. In Figure 3·12, both Actions c and e wake up some sleeping servers. Action c is called due to a "QoS crisis" (i.e., QoS of jobs in some clusters tends to violate the SLAs), and the number of idle servers at time t is smaller than the number of required servers. Therefore, additional servers are required to be waken up in order to increase QoS. The number of servers to be waken up, i.e., $N_{up}(t)$ is determined as:

   $$N_{up}(t) = min\big\{N_{req}(t) - N_{idle}(t) - f_{eqv}(N_{tran}(t)), N_{slp}(t)\big\}, \tag{3.37}$$

   where $N_{slp}(t)$ is the number of servers in sleep state at t. Here we also take the number of servers in the transition state, i.e., $N_{tran}(t)$, into account by a function $f_{eqv}(\cdot)$. $f_{eqv}(\cdot)$ calculates the equivalent number of idle servers that the number of servers in transition is, based on the time that these servers have spent in the transition state. Specifically, assuming that a server s has been in the transition state for $T_{w,s}(t)$ seconds at time t, and the waking up process takes T_{res} seconds introduced in Section 3.2.1, then this server s is equivalent to $T_{w,s}(t)/T_{res}$ idle server. Hence,

   $$f_{eqv}(N_{tran}(t)) = \frac{\sum_{s=1}^{N_{tran}(t)} T_{w,s}(t)}{T_{res}}. \tag{3.38}$$

   In Action e, $N_{up}(t)$ is determined for the purpose of achieving better signal tracking. Waking up a server can increase its power from P_{slp} to P_{tran}. Hence:

$$N_{up}(t) = min\{\frac{P_{tgt}(t+4) - P_{con}(t)}{P_{tran} - P_{slp}}, N_{slp}(t)\}, \tag{3.39}$$

where P_{tran} and P_{slp} are the power consumption of the server in the transition state and in the sleep state introduced in Section 3.2.1, respectively.

2. Put idle servers into sleep. In Action g, the number of idle servers to be put into sleep state, i.e., $N_{toslp}(t)$, is determined for the purpose of achieving better signal tracking, as putting an idle server into the sleep state can reduce its power from P_{idle} to P_{slp}. Therefore,

   $$N_{toslp}(t) = min\{\frac{P_{con}(t) - P_{tgt}(t+4)}{P_{idle} - P_{slp}}, N_{rdslp}(t)\}, \tag{3.40}$$

   where P_{idle} is the server idle power, $N_{rdslp}(t)$ is the number of servers that are ready to be put into the sleep state at time t. $N_{rdslp}(t)$ is introduced to avoid waking up servers or putting servers into sleep over frequently. Here we apply the similar timeout mechanism that is used in the best tracking policy in Section 3.4, in which only when a server has been in the idle state for the time longer than T_{out} introduced in Eq. (3.5), it can be put into the sleep state. We use $T_{idle,s}(t)$ to denote the time period that the server s has been in the idle state till time t. If $T_{idle,s}(t) \geq T_{out}$, then the server s is ready to be put into the sleep state. Therefore:

   $$N_{rdslp}(t) = \sum_{s=1}^{N_{idle}(t)} \mathbb{I}_{\{T_{idle,s}(t) \geq T_{out}\}}. \tag{3.41}$$

3. Assign idle servers to clusters to serve waiting jobs. If there are both waiting jobs and idle servers in the system, we can assign idle servers to clusters to

serve some of the waiting jobs[9]. Since all M clusters share the idle server pool, the number of idle servers assigned to each cluster j, i.e., $N_{assg,j}(t)$ needs to be determined. The first step is to determine the total number of idle servers that are going to be activated and assigned, i.e. $N_{assg}(t)$. In Figure 3·12, Actions a and b are called because of the "QoS crisis", hence $N_{assg}(t)$ is calculated based the QoS requirement. In Action a, since the number of idle servers is not smaller than the number of required servers, i.e., $N_{idle}(t) \geq N_{req}(t)$, we simply activate and assign $N_{req}(t)$ servers to clusters, i.e., $N_{assg}(t) = N_{req}(t)$. In Action b, since $N_{idle}(t) < N_{req}(t)$, i.e., the idle servers are not sufficient, thus all the idle servers at time t require to be activated, i.e., $N_{assg}(t) = N_{idle}(t)$.

Differing from Actions a and b, In Action f, $N_{assg}(t)$ is determined for the purpose of achieving better signal tracking, hence $N_{assg}(t)$ is calculated as:

$$N_{assg}(t) = min\Big\{\frac{P_{tgt}(t+4) - P_{con}(t)}{P_{max} - P_{idle}},\ N_{idle}(t),\ q(t)\Big\},$$

where P_{max} is the server maximal power, $q(t)$ is the total number of queued jobs in the whole system at time t, i.e., $q(t) = \sum_{j=1}^{M} q_j(t)$. The equation represents that the number of activated servers must be smaller than the current number of idle servers $N_{idle}(t)$ and the number of queued jobs $q(t)$. Note that in Actions a and b, $q(t)$ has been considered in the calculation of $N_{req}(t)$ in Eq. (3.35), hence we do not need to consider it separately here. Overall we have:

$$N_{assg}(t) = \begin{cases} N_{req}(t), & \text{in Action } a, \\ N_{idle}(t), & \text{in Action } b, \\ min\big\{\frac{P_{tgt}(t+4)-P_{con}(t)}{P_{max}-P_{idle}},\ N_{idle}(t),\ q(t)\big\}, & \text{in Action } f. \end{cases} \tag{3.42}$$

---

[9] If the signal tracking has a higher priority than the workload QoS and the power cap determined by our policy is not sufficient to support additional active servers, even if there are both waiting jobs and idle servers in the system, idle servers are not activated for job servicing.

Next, we determine the number of idle servers assigned to each cluster j. First, the number of assigned servers to each cluster j, i.e., $N_{assg,j}(t)$ cannot be larger than the number of jobs waiting in the queue in that cluster, i.e., $q_j(t)$:

$$\begin{aligned} &\textstyle\sum_{j=1}^{M} N_{assg,j}(t) = N_{assg}(t), \\ &N_{assg,j}(t) \leq q_j(t),\ \forall j. \end{aligned} \tag{3.43}$$

There are many different ways to allocate idle servers to each cluster. Simple methods of the allocation include round robin, random, or shortest job first. From experiments, we find that an "urgent QoS first" strategy that allocates idle servers based on workload QoS and its constraints can provide better performance in both QoS guarantee and signal tracking than other popular methods in our scenario. Specifically, the "urgent QoS first" strategy is as follows: the mean of the QoS degradation of each cluster j in the past time window $[t-T, t]$, i.e., $\bar{D}_j(t)$ is first calculated as introduced in Eq. (3.29); then we calculate the weight for each cluster:

$$w_j(t) = \frac{\bar{D}_j(t)}{Q_j}, \forall j \in 1, 2, ...M. \tag{3.44}$$

We assign $N_{assg}(t)$ idle servers to each cluster based on the sorted order of $w_j(t)$. The larger $w_j(t)$ represents a more "urgent" scenario in terms of satisfying the SLAs, and thus clusters with larger $w_j(t)$ are given higher priorities in receiving idle servers. The cluster with the largest $w_j(t)$ first receives its required number of servers $N_{req,j}(t)$, followed by the cluster with the second largest $w_j(t)$, and so on. Note that clusters with lower priorities may not receive sufficient servers at t to meet with their requirements $N_{req,j}(t)$, but since the QoS feedback $\bar{D}_j(t)$ is updated dynamically at every time interval, these clusters would get higher priorities later, and receive sufficient servers to meet their SLAs in the long run.

In Action f, it is possible that after satisfying $N_{req,j}(t)$ for every cluster j, there are still idle servers required to be assigned in order to better track the RSR signal. In this case, these additional activated idle servers that are only for the signal tracking purpose and have no constraints on QoS are allocated to clusters in a round robin manner to keep fairness.

4. Regulate the power and service rate of active servers. By default all servers are running at their maximal service rate for the purpose of overall energy conservation and QoS guarantee. However, sometimes in order to better track the signal, servers are required to be slowed down when the signal power cap is low, i.e., in Action d, as decreasing the service rate can reduce the server power, which has been introduced in Section 3.2.1.

   Since each cluster runs a homogeneous set of jobs, we have assumed that all servers in each cluster j are always running at the same service rate $u_j(t)$, and thus the same power $P_j(t)$. We determine $u_j(t)$ by considering both signal tracking and QoS requirements. To guarantee QoS, we set a minimal bound of $u_j(t)$ for each cluster j at time t, i.e., $u_{min,j}(t)$, based on Eq. (3.30) and Eq. (3.32), i.e.,:

   $$u_{min,j}(t) = u_{max,j} \cdot min\{1, \frac{\bar{D}_j(t)+1}{\beta Q_j + 1}\}. \tag{3.45}$$

   Eq. (3.45) represents that if the current average job system time in cluster j satisfies the constraint in Eq.(3.32), i.e., $\bar{T}_{sys,j}(t) = (\bar{D}_j(t)+1) \cdot T_{min,j} < (\beta Q_j + 1) \cdot T_{min,j}$, then there is room to slow down active servers from the default maximal service rate $u_{max,j}$, otherwise, all active servers must run at their maximal service rate. This minimal bound on the service rate also prevents jobs stalling in the server forever.

We then calculate the total maximal possible reduction in data center power consumption at t based on Eq.(3.1) and (3.45) as:

$$P_{max,rd}(t) = \sum_{j=1}^{M} N_j(t) \cdot k_j\big(u_{max,j} - u_{min,j}(t)\big). \tag{3.46}$$

We define a coefficient $\delta(t)$ as:

$$\delta(t) = max\big\{1, \frac{P_{con}(t) - P_{tgt}(t+4)}{P_{max,rd}(t)}\big\}, \tag{3.47}$$

where $P_{con}(t) - P_{tgt}(t+4)$ is the required amount of data center power reduction in order to perfectly track the signal. Then the service rate $u_j(t)$ for each cluster j is:

$$u_j(t) = u_{max,j} - \delta(t) \cdot \big(u_{max,j} - u_{min,j}(t)\big). \tag{3.48}$$

In this way, the power reduction is fairly distributed to clusters based on their minimal bounds in service rate.

3.7.2 Energy and Reserve Bidding

With the proposed EnergyQARE runtime policy introduced in the prior section, in this section we investigate the optimal power and reserve bidding strategies. Similar to the optimization problem formulated in the single server RSR provision in Section 3.3.1 and Eq. (3.4), the problem on data center RSR provision here is to minimize the data center energy monetary costs under the constraints of both the tracking error and the workload QoS introduced in Eq. (2.2) and Eq. (3.3), respectively. We conduct similar studies in the sample frequency distributions on the 1-hour trajectories of the tracking error $\epsilon(t)$ and the QoS degradation of jobs i in each cluster j (i.e., $D_{i,j}$) for a sufficiently large number of simulations, as what we did in the single server case, and find that they also fit the Gamma distribution $\Gamma(k, \theta)$ with shape k:

$k_\epsilon = \bar{\epsilon}^2/\sigma_\epsilon^2$, $k_{D_j} = \bar{D}_j^2/\sigma_{D_j}^2$ and scale θ: $\theta_\epsilon = \sigma_\epsilon^2/\bar{\epsilon}$, $\theta_{D_j} = \sigma_{D_j}^2/\bar{D}_j$, where $\bar{\epsilon}$, σ_ϵ, $\bar{D}_j$, σ_{D_j} are mean and standard deviation of the tracking error and QoS degradation of jobs in cluster j during the hour. Then the optimization problem is formulated as:

$$\begin{aligned} \underset{\bar{P},R,\ \text{policy}}{\text{minimize}} \quad & \Pi^E \cdot \bar{P} - \Pi^R \cdot R + \Pi^\epsilon \cdot \bar{\epsilon} \\ \text{subject to} \quad & \Gamma(k_\epsilon, \theta_\epsilon, \epsilon^{tol}) \geq \eta^\epsilon, \\ & \Gamma(k_{D_j}, \theta_{D_j}, Q_j) \geq \eta_j,\ \forall j, \\ & \bar{P} + R \leq N \cdot P_{max}, \\ & \bar{P} - R \geq N \cdot P_{slp}, \\ & \bar{P} \geq 0, R \geq 0, \end{aligned} \tag{3.49}$$

recall that N is the total number of servers in the data center, P_{max} and P_{slp} are the maximal possible server power and power of sleep state, respectively. $N \cdot P_{max}$ and $N \cdot P_{slp}$ are maximal and minimal possible power consumption of the data center. We assume the given data center runs the EnergyQARE runtime policy introduced in Section 3.7.1 as the “policy”, and solve the optimal bidding value $(\bar{P}, R)$ accordingly.

Due to the complexity of the problem, instead of applying analytical methods, we solve the optimal solution using numerical methods. We conduct exhaustive search $(\bar{P}, R)$ over a sufficient fine granularity. For each pair of $(\bar{P}, R)$, we first simulate a 1-hour RSR provision period multiple times with varying signals and workload arrival, so as to estimate the mean and standard deviation statistics of $\epsilon(t)$ and $D_{i,j}$. Then we apply these statistics to Eq. (3.49) and search for the optimal $(\bar{P}, R)$. An alternative solution is to build regression models on statistics of the tracking error and the QoS degradation via $(\bar{P}, R)$ through extensive simulations, and leverage these regression models to solve Eq. (3.49).

Table 3.7: Workload properties in experiments - Trace 1.

Application Name j	Shortest Runtime $T_{min,j}$ (sec)	Arrival Rate λ_j (# of jobs / sec)	QoS Degradation Tolerance Q_j
Canneal	44.3	0.11	0.3
Bodytrack	51.0	0.10	2.0
Ferret	74.3	0.20	1.0
Freqmine	95.2	0.11	0.1
Facesim	149.6	0.10	0.5

Table 3.8: Workload properties in experiments - Trace 2.

Application Name j	Shortest Runtime $T_{min,j}$ (sec)	Arrival Rate λ_j (# of jobs / sec)	QoS Degradation Tolerance Q_j
Blackscholes	23.5	0.43	1.0
Vips	24.5	0.20	0.7
Raytrace	42.4	0.07	0.8
Dedup	58.8	0.14	0.3
Ferret	74.3	0.07	0.2
Fluidanimate	93.3	0.14	2.0
Steamcluster	151.2	0.05	0.5

3.7.3 Experimental Results

In this section, we simulate data center RSR provision with EnergyQARE in different scenarios, and evaluate the signal tracking performance, workload QoS, and the data center energy monetary savings.

Methodology

We simulate the (heterogeneous) workload arrival as the Poisson process (i.e., the arrival time interval follows exponential distribution) with Monte Carlo simulation methods. Different applications are randomly selected from the PARSEC-2.1 benchmark suite in simulation. The power-throughput profiles of each application are taken from real-life measurements, as introduced in Section 3.2.1. The workload arrival rate is calculated based on the size of the data center, i.e., the total number of servers N

Table 3.9: Workload properties in experiments - Trace 3.

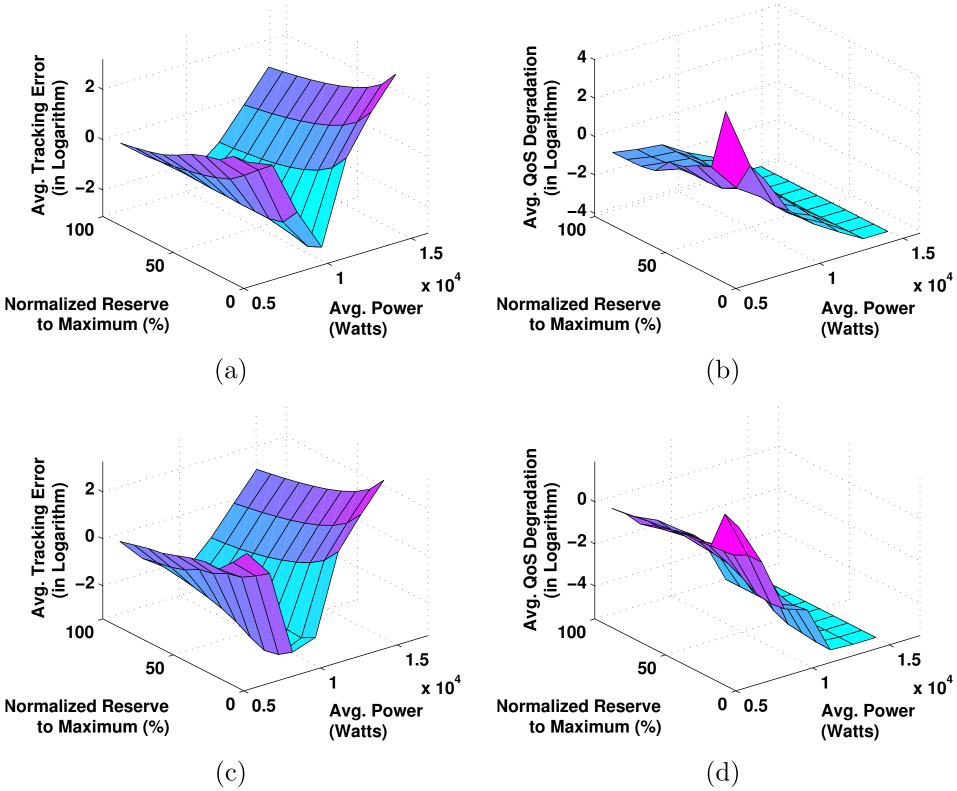
Application Name j	Shortest Runtime $T_{min,j}$ (sec)	Arrival Rate λ_j (# of jobs / sec)	QoS Degradation Tolerance Q_j
Raytrace	42.4	0.12	0.5
Canneal	44.3	0.29	0.2
Bodytrack	51.0	0.25	2.0
Swaptions	74.0	0.07	0.8
Fluidanimate	93.3	0.11	1.0
Steamcluster	151.2	0.03	0.4

in the data center, and the utilization U, i.e., the percentage of servers that are in active state on average. By default, we use the data center consisting of 100 servers and at 50% utilization. Results of different data center sizes and utilization are evaluated in case studies. We use the shallow sleep ($T_{res} = 10s$, $P_{slp} = 10\%P_{max}$) as the default sleep state. We simulate a 1-hour period experiment multiple times with different RSR signal and workload arrival traces, and evaluate the signal tracking, QoS performance, and the energy monetary savings.

Signal Tracking Performance and Workload QoS

We first conduct experiments on the default settings with multiple random workload arrival traces. In each workload arrival trace, the types of application j that are contained, the arrival rate of each of them, λ_j, and the QoS tolerance in SLAs, i.e., Q_j are randomly generated. All these workload arrival traces are constrained to have utilization at $U = 50\%$. To better evaluate the capability of our policy in guaranteeing SLAs, we set some Q_j to small values (shown in Table 3.7, Table 3.8 and Table 3.9), so that some SLAs are tight and easy to be violated with random policies. η_j is set to 90% in all SLAs. The tracking error probabilistic constraint is set as $(\epsilon^{tol}, \eta^{\epsilon}) = (0.3, 90\%)$ based on today's market information (PJM, 2016).

Since different workload arrival traces lead to varying results (though the differ-



(a)

(b)

(c)

(d)

Figure 3·13: Mean of the signal tracking error and the QoS degradation via different $\bar{P}$ and R (normalized to the maximal possible value R_{max} given $\bar{P}$) in EnergyQARE. (a) and (b) are results of the workload trace 1; (c) and (d) are results of the workload trace 2. (b) is for the Canneal application in trace 1 and (d) is for the Dedup application in trace 2.

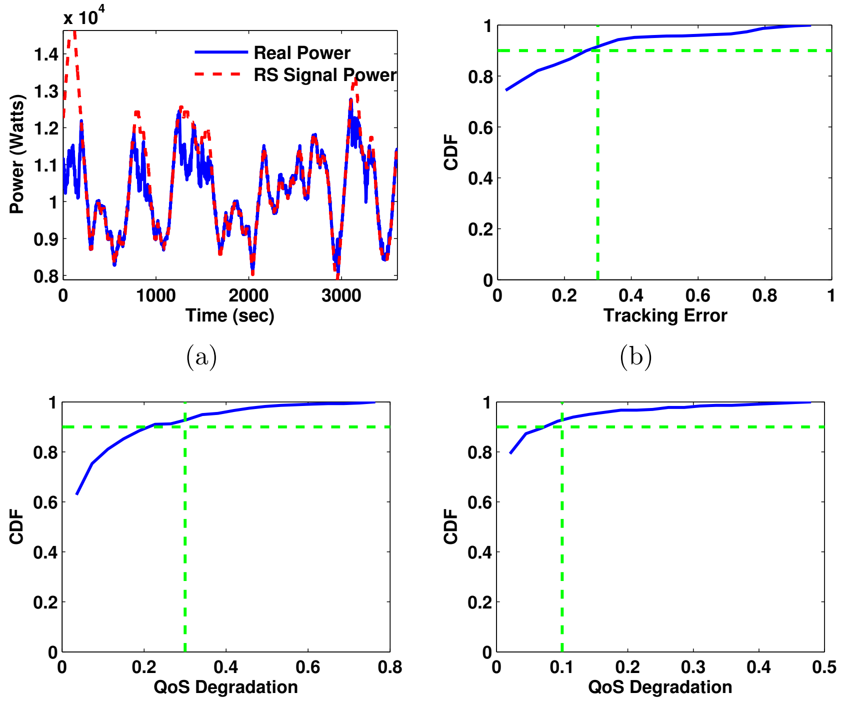
ences are small from the observation in experiments), in order to achieve generalized results, we test on multiple different workload arrival traces, and evaluate the statistics of results. In this section, we randomly select three of them to demonstrate the results. Properties of three selected workload traces are listed in Table 3.7, Table 3.8 and Table 3.9.

Figure 3·13 shows the signal tracking performance and the workload QoS via different pairs of bidding values $(\bar{P}, R)$, where R is normalized to its maximal possible value R_{max} given $\bar{P}$, i.e., $R_{max} = min\{N \cdot P_{max} - \bar{P}, \bar{P} - N \cdot P_{slp}\}$. Figure 3·13(a) is the average tracking error during the 1-hour experiment via $(\bar{P}, R)$ pairs for the workload trace 1. From the figure, tracking error is more sensitive to the average power $\bar{P}$ than the reserve R. The average of the signal tracking error is large when the average power $\bar{P}$ is either low or high; in other words, the best tracking performance appears in the middle of the range of $\bar{P}$. When $\bar{P}$ is low, the overall power budget to the data center is insufficient to guarantee workload SLAs. As a result, in order to satisfy the SLAs, the signal power cap is frequently violated, resulting in low signal tracking accuracy. When $\bar{P}$ is high, workloads are fast served and queues are often empty. Then servers are frequently in the idle or sleep states, in which the power cannot be regulated, leading to poor signal tracking performance.

Figure 3·13(b) is the average QoS degradation of the Canneal application in workload trace 1. From the figure, the QoS degradation is also more sensitive to $\bar{P}$ than R. However, unlike the tracking error, the QoS degradation has a monotonic relation to $\bar{P}$: the higher $\bar{P}$ is, the smaller the QoS degradation will be. When $\bar{P}$ is high, the power budget is sufficient to run workloads faster. Similar results are found in the QoS degradation of all the other applications in trace 1. In addition, we notice that the QoS is more sensitive to R when $\bar{P}$ is low. When $\bar{P}$ is low, a higher R provides larger range in power and more flexibilities in power budgeting and workload servicing, and thus leads to better QoS performance.

Figure 3·13(c) and Figure 3·13(d) are results of the tracking error and the workload QoS via different pairs of $(\bar{P}, R)$ for the workload trace 2. These results are similar to the case of workload trace 1. Similar results are also found in not only workload trace 3, but also all the other workload traces we have tested in experiments.

Next, we evaluate the signal tracking and the QoS in the runtime given $(\bar{P}, R)$ at the optimal values in Figure 3·14 (for workload trace 1). Figure 3·14(a) visualizes the real power consumption $P_{con}(t)$ and the RSR signal power cap $P_{tgt}(t)$ dynamically



(a) (b)

(c) (d)

Figure 3·14: Results of RSR signal tracking and QoS degradation for the workload trace 1 with $(\bar{P}, R)$ at their optimal values in EnergyQARE. (a) is the real dynamic power $P_{con}(t)$ compared with the signal power cap $P_{tgt}(t)$ during the 1-hour simulation. (b) is the cumulative distribution function (CDF) of the tracking error $\epsilon(t)$. The green lines show the tracking error probabilistic constraints, i.e., $(\epsilon^{tol}, \eta^{\epsilon})$. (c) and (d) are the CDF of the QoS degradation of Canneal and Freqmine applications, respectively. The green lines are the SLAs, i.e., (Q_j, η_j).

for the 1-hour trace. In most of time, $P_{tgt}(t)$ is well tracked by $P_{con}(t)$, with only a few of notable violations when the signal values are high, and there are no sufficient workloads in the system. In fact, from the experiments, the tracking accuracy can be further increased if a lower $\bar{P}$ is selected, however, the workload SLAs are then violated as a side effect. Our solution can better handle the trade-off between signal tracking and QoS.

Figure 3·14(b) is the cumulative distribution function (CDF) of the tracking error $\epsilon(t)$ during the 1-hour simulation. The green lines represent the probabilistic con-

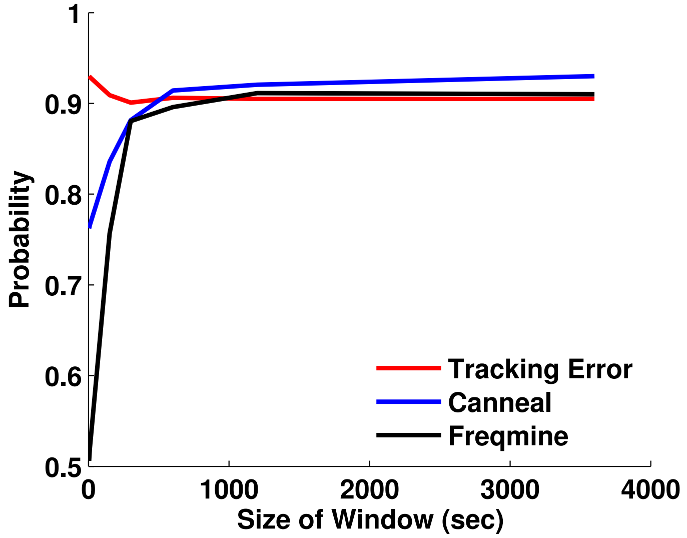


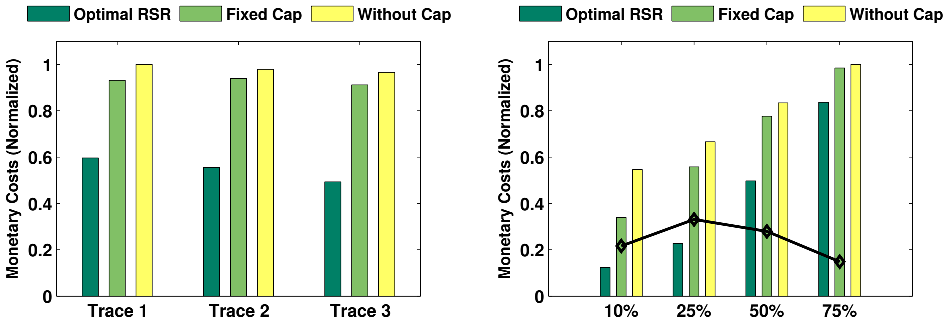
Figure 3·15: The impact of the size of the feedback window T to the tracking error and workload QoS in EnergyQARE. The probabilities of the tracking error and the QoS degradation that are smaller than ϵ^{tol}, and Q_j for Canneal and Freqmine applications respectively, are shown in the figure.

straints $(\epsilon^{tol}, \eta^{\epsilon})$. Figure 3·14(c) and 3·14(d) are the CDF of the QoS degradation of two selected applications from workload trace 1 with the tightest SLA constraints, i.e.,: the Canneal and the Freqmine applications (see in Table 3.7). The green lines represent the SLA probabilistic constraints (Q_j, η_j). Tracking error and workload QoS all meet the constraints from the figures. The rest of three applications (not shown here) in workload trace 1 have even better QoS (i.e., higher probabilities η_j in satisfying the tolerances Q_j) due to their loosen SLA constraints. Moreover, all the other tested traces show similar results to these figures. Overall, EnergyQARE enables the data center to participate in the RSR provision with accurate signal tracking, while also guaranteeing workload QoS.

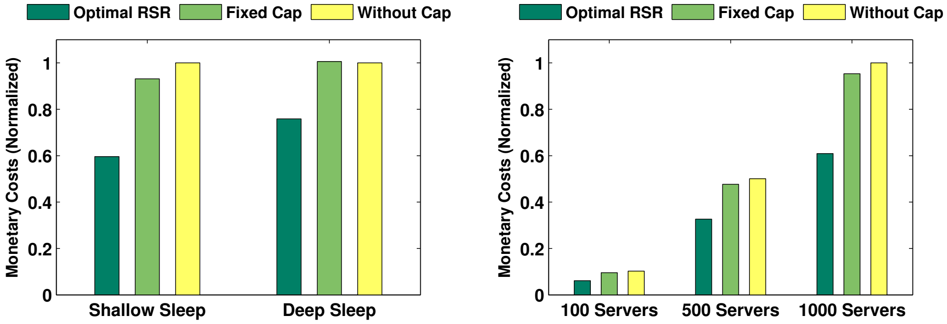
Figure 3·15 shows the impact of different sizes T of the feedback window (described in Eq. (3.28)) on the tracking error and workload QoS. The probabilities of the tracking error and QoS degradation (of both Canneal and Freqmine) that are smaller than ϵ^{tol} and Q_j respectively are shown in the figure. From the figure, the QoS is poor with a small window size. When the window size is small, the policy makes decisions only based on recent observation of the performance, hence decisions are with higher variances. Since the system time of applications in simulation varies from a few seconds to minutes, short observation in a small window fails to effectively characterize the overall workload QoS, and thus leads to inaccurate feedback and poor decisions. In terms of the signal tracking, however, the policy with a small window size tends to make decisions based on the instantaneous tracking error, which is similar to what a best tracking policy (introduced in Section 3.4) does, and therefore leads to better tracking performance. In addition, the figure also demonstrates that the tracking performance and workload QoS are both getting to constants after the window size increases and reaches to a certain value.

Energy Monetary Savings

Next, we evaluate the energy monetary savings of data center participation in RSR. In Figure 3·16(a), we compare the optimal monetary costs of data centers with RSR provision by EnergyQARE, i.e., "optimal RSR", to the "fixed cap" scenario and the "without cap" scenario, for workload traces 1, 2 and 3, respectively. The "fixed cap" scenario represents that the data center consumes the average power at $\bar{P}$ (so the overall energy consumption is close to that of "optimal RSR"), but with no reserve provision, i.e., $R = 0$. The "without cap" scenario represents that the data center serves workloads without any constraints on the power consumption. The energy monetary costs in "fixed cap" and "without cap" scenarios are calculated simply based on the total energy consumed, i.e., $\Pi^E \cdot E$, where E is the total energy consumed during the simulation period (i.e., 1-hour). To better evaluate the savings, we normalize the energy monetary cost of each scenario to the largest value in all



(a) Different Workload Traces (b) Different Utilization



(c) Different Sleep States (d) Different Data Center Sizes

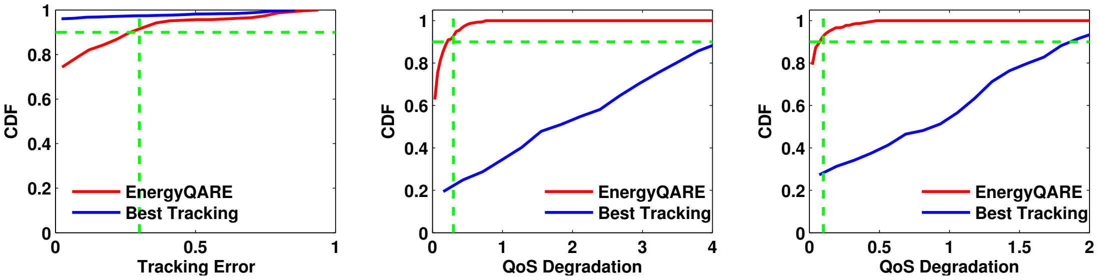
Figure 3·16: The energy monetary costs in three scenarios: “optimal RSR”, ”fixed cap” and “without cap” in different cases. All the costs are normalized to the largest value in each figure. In (b) we also calculate the absolute value of the cost *savings* from the “optimal RSR” scenarios to their corresponding “fixed cap” scenarios, and represent the absolute savings in the black line.

scenarios demonstrated in each figure[10]. We use $\Pi^E = \Pi^R = \Pi^\epsilon = 0.1\$/kWh$ based on today’s markets (PJM, 2016; Aikema et al., 2012).

From Figure 3·16(a), savings in different workload traces are similar: the extensive simulations on different workload arrival traces demonstrate a less than 5% variation in energy monetary savings. Providing RSR with EnergyQARE, the data center saves on average 41% energy cost compared to the “fixed cap” scenario, and 44% compared to the “without cap” scenario, while also satisfying the workload SLAs.

---

[10] The largest cost is most likely to appear in one of the “without cap” scenarios in each figure.



(a) CDF of Tracking Error (b) QoS CDF of Canneal (c) QoS CDF of Freqmine

Figure 3·17: The CDF of tracking error and QoS degradation of both the EnergyQARE and the best tracking policy for workload trace 1. The blue curves represent the best tracking policy, and the red curves represent the EnergyQARE. The green lines represent the tracking error and the workload SLA probabilistic constraints, i.e., $(\epsilon^{tol}, \eta^{\epsilon})$ and (Q_j, η_j).

Comparison of Policies

We then compare the results of EnergyQARE with the prior introduced best tracking policy in Section 3.4. The best tracking policy simply regulates the data center power dynamically to track the instantaneous RSR signal as accurate as possible, but does not explicitly consider workload QoS.

Figure 3·17 shows the results of the EnergyQARE and the best tracking policy with workload trace 1. The CDF of the tracking error, and the CDF of the QoS degradation of Canneal and Freqmine are shown in Figure 3·17(a), 3·17(b) and 3·17(c), respectively. Green lines represent the probabilistic constraints on the tracking error and the SLAs, i.e., $(\epsilon^{tol}, \eta^{\epsilon})$ and (Q_j, η_j). From the figure, though the best tracking policy provides better tracking performance in Figure 3·17(a), because it minimizes the instantaneous tracking error, the workload QoS of the policy is much worse than that of the EnergyQARE based on Figure 3·17(b) and Figure 3·17(c). The workload SLAs are violated by the best tracking policy in both of the figures, due to the fact that the best tracking policy does not take QoS feedback into account in decision making.

The EnergyQARE, on the other hand, satisfies all signal tracking and workload SLA constraints.

Case Study 1: Multiple Utilization Settings

We now study how the utilization of the data center impacts the performance and the savings in RSR provision. We simulate the workload arrival traces using the same type of applications and the same SLAs to workload trace 1, but with different utilization as: 10%, 25%, 50% and 75%. All information in Table 3.7 remains the same, except for the arrival rates of applications, which are scaled up for the 75% utilization scenario and scaled down for the 10% and 25% utilization scenarios. The same RSR signal trace is used in all scenarios.

Figure 3·16(b) shows the data center energy monetary costs of the “optimal RSR”, the “fixed cap”, and the “without cap” scenarios in different utilization settings. Costs in all the scenarios and settings are normalized to the maximal value in the figure. For 10%, 25%, 50% and 75% utilization cases, the savings from the “optimal RSR” to the “fixed cap” scenario are 63.7%, 59.3%, 36.0% and 15.1%, and to the “without cap” scenario are 77.4%, 65.9%, 40.4% and 16.4%, respectively. We see that the percentage of savings increases when the utilization decreases, though it is getting saturated when the utilization further decreases. We also calculate the absolute values of the savings from the “optimal RSR” scenarios to the corresponding “fixed cap” scenarios for all utilization settings, and present the results in the black line in Figure 3·16(b). We notice that the largest absolute value of the monetary saving (i.e., corresponding to the largest amount of reserve provision) appears in the middle level of the utilization. When the utilization is high, the data center is busy with job servicing, and has little flexibility in regulating the power and tracking the signal. When the utilization is low, the data center does not have sufficient jobs to be served, which limits the number of servers that can be activated, and thus limits the power regulation capability of the data center. Therefore, the amount of RSR provision is small when the data center utilization is either too high or too low.

Case Study 2: Shallow Sleep vs. Deep Sleep

As introduced before, many servers in today's data centers support different sleep states. So far by default we have used a shallow sleep state that has relatively high sleep power but short transition delay. In this section, we experiment on another sleep state, the deep sleep state that has longer transition period for waking up, but lower sleep power. We compare the results of shallow and deep sleep states. Parameters of different sleep states used in this case study are introduced in Section 3.2.1. We apply the workload trace 1 and the same RSR signal trace to both cases for fair comparison.

Figure 3·16(c) shows the monetary costs in the "optimal RSR", the "fixed cap", and the "without cap" scenarios using the shallow sleep and the deep sleep state, respectively. Using the shallow sleep, the savings from the "optimal RSR" to the "fixed cap" and the "without cap" are 36.0% and 40.4%, respectively, while using the deep sleep, both numbers drop to 24%. Since RSR provision requires fast and frequent data center power regulation to track the fast changed signal, the shallow sleep state is more suitable to be used due to the short delay in state transition, and therefore leads to higher savings than the deep sleep state. Such results well match with our previous observation in the best tracking policy in Section 3.4. In addition, we find that with a deep sleep state, having a fixed power cap (i.e., "fixed cap") does not provide energy cost savings compared to "without cap" from the figure. When having a power cap, servers are frequently forced to be put into sleep in order to meet the cap, and waken up later once needed. The deep sleep state leads to tremendous time delay and energy loss during these frequent transitions. Therefore, having a power cap on servers with deep sleep states is not an efficient strategy in terms of the energy cost savings.

Case Study 3: Scalability via Data Center Size

Finally, we study the scalability of the performance and savings via the size of the data center, i.e., the number of servers in it. We simulate the workload arrivals based on workload trace 1, at 50% utilization with different number of servers. All information in Table 3.7 remains the same, except for the arrival rates, which are scaled up and down based on the number of servers. Still, we use the same RSR signal trace in simulation. Figure 3·16(d) shows the energy monetary costs for data centers containing 100, 500 and 1000 servers. From the figure, we see similar percentage of savings from RSR provision in all three cases, which demonstrates that the savings are scalable via the size of data center.

3.8 A Prototype Implementation of Data Center RSR

So far we have mainly modeled and simulated the data center RSR provision. Being able to demonstrate the capabilities and benefits of RSR provision on the real data center clusters would further achieve significant impacts on industrial society. In this section, we design a prototype of data center RSR provision on a real-life system - a real multi-core server. Specifically, we implement our optimization framework and the runtime policy (for the single server, introduced in Section 3.3) on a multi-core virtualized server, i.e., a 1U server that has an AMD mangy Cours (Opteron 6172) processor, with 12 cores on a single chip, virtualized by VMware vSphere 5.1 ESXi hypervisor, leveraging the CPU resource limits control knob. We test the capabilities of the real server power capping technique in tracking the RSR signal. This initial test can provide guidance for the future deployment of our techniques onto real-life data centers for practical uses.

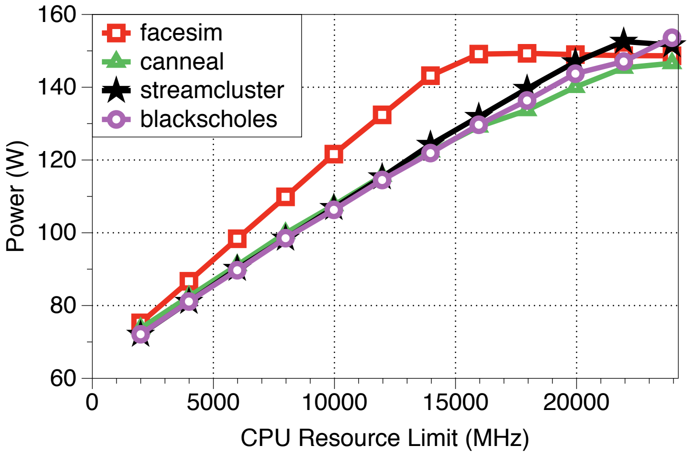


Figure 3·18: Power- CPU resource limits models for the PARSEC applications.

3.8.1 Real Server RSR Provision Capability Test

In practice, before issuing a contract to an RSR provision candidate, ISO first gives the candidate a test signal to track and examines its performance mainly on three aspects: (1) the capability of consuming a stable power for a period of time (i.e., 5 minutes); (2) the time required for power consumption to ramp up to $\bar{P} + R$ and down to $\bar{P} - R$; (3) the capability of making dynamic power changes at a sufficiently fine granularity (PJM, 2013). Our experimental results on the 1U server show the following:

- *Power Stability*: We keep the resource limit at a fixed setting for 10 minutes and observe the fluctuation of the power consumption. The standard deviation of the power consumption when a given PARSEC application is in its parallel phase is 1-3W, which is only 1-2% of $\bar{P}$;

- *Ramp-up Capability*: Our server shows the ability to ramp up to 153W and down to 66W (66W is the server idle power, 153W is the maximal power consumption of Blackscholes) at 1s interval;

- *Granularity of Modulation*: The resource limit control knob is able to modulate the power consumption in a granularity of a few milliwatts.

These results show that our server, a typical virtualized server in data centers, can meet all the ISO test requirements using the CPU resource limits control knob, and it has sufficient capability for providing RSRs.

3.8.2 Power - Resource Limits Model

Since we use the CPU resource limits as the power capping control knob, the relation between CPU resource limits and power consumption should be studied. Figure 3·18 shows the power consumption of four PARSEC applications as a function of the CPU resource limits. For most of the applications, power and CPU resource limits are linearly correlated; therefore, the peak power consumption is observed at the highest CPU resource limit. However, for applications such as Facesim, power consumption is constant after reaching a certain amount of CPU resource, as the application cannot continue to utilize the resource efficiently.

To capture the various relationships between power and CPU resource limits, we monitor the overall CPU usage (in MHz) (i.e., CPU_{used}) and the amount of CPU resource that is not utilized because of the existing CPU resource limits on the system (i.e., CPU_{ready}). VM statistics (i.e., CPU_{used} and CPU_{ready}) are polled every 2 seconds using the vSphere SDK for GuestOS library. CPU_{ready} metric allows us to capture the saturating performance effects (e.g., Facesim), as it reflects the amount of CPU resource needed to reach the maximal performance. The CPU_{used} value captures the utilization levels.

We also use feedback from the power meter to update the power/CPU resource limits models dynamically at runtime. In other words, power measurements (i.e., P_{con}) are fed into the power-CPU resource limit model to estimate the CPU resource

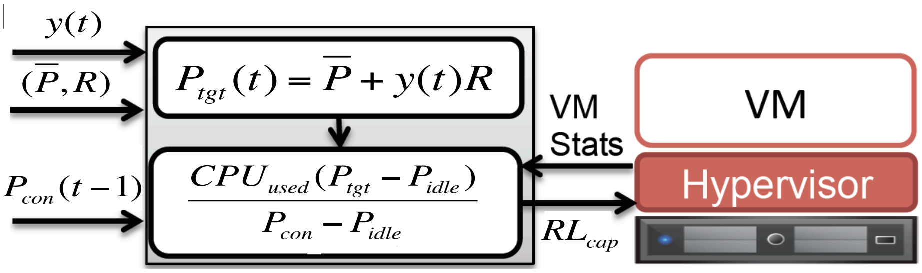

$y(t)$
$(\overline{P}, R)$
$P_{con}(t-1)$
$P_{tgt}(t) = \overline{P} + y(t)R$
$\frac{CPU_{used}(P_{tgt} - P_{idle})}{P_{con} - P_{idle}}$
VM Stats
RL_{cap}
VM
Hypervisor

Figure 3·19: The overview of the runtime power capping technique. Our technique receives input from the ISO and the VM (e.g., CPU_{used}, etc.) to make CPU resource limit adjustments so as to keep the power consumption close to the current power cap.

limit that meets the power cap. We first remove the idle power consumption from the measured total power to compute the dynamic power, $P_{dyn} = P_{con} - P_{idle}$. In our runtime policy, we estimate the CPU resource limit value, i.e., RL_{cap} that meets the given power cap by using the Eq. (3.50), where P_{tgt} is the power cap value. From our experiments, the estimation error of this model is less than 5%:

$$RL_{cap}(MHz) = \frac{CPU_{used} \cdot (P_{tgt} - P_{idle})}{P_{con} - P_{idle}}. \tag{3.50}$$

3.8.3 Runtime Control Knobs and Policy Implementation

The goal of the runtime policy is to track the RSR signal on a virtualized server environment. Figure 3·19 shows the overview of our implementation. Our power capping technique receives three inputs: (1) $(\bar{P}, R)$ from the optimization engine, (2) the real-time power measurements (i.e., $P_{con}(t-1)$) from the power meter and (3) CPU resource usage statistics from the VM. The output of the power capping module is the CPU resource limit that is expected to keep the power consumption of the server close to the current power cap $P_{tgt}(t)$.

After receiving the RSR signal $y(t)$, we derive the corresponding power cap value,

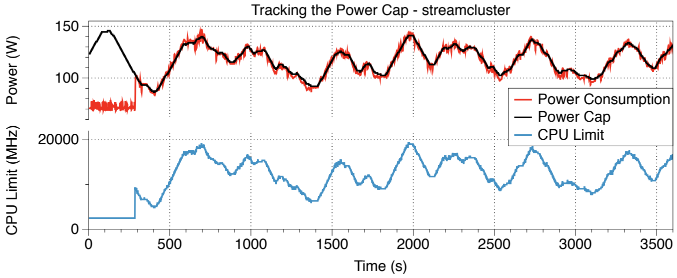


Figure 3·20: 1-hour RSR signal power capping of Streamcluster application by adjusting the CPU resource limits. We both show the real power consumption and the power cap values (top figure) and the dynamic adjustment of CPU resources (bottom figure).

i.e., $P_{tgt}(t) = \bar{P} + y(t)R$. We then calculate the CPU resource limit that matches the P_{tgt}. After calculating the CPU resource limit, our policy communicates with the ESXi host to reconfigure the VM CPU resource limits by using the vSphere SDK library. We monitor the power consumption and the power cap every second, and adjust the CPU resource limits if the average absolute tracking error over the last 2 seconds is larger than 2W.

3.8.4 Real Server RSR Signal Tracking Performance

We first investigate the homogeneous workload case. Figure 3·20 shows 1-hour long power profile for Streamcluster application. The result shows that except for the idle period at the very beginning, during which time we are unable to tune the power by resource limits control knobs, our runtime policy enables our server to dynamically track the RSR power cap accurately. The average tracking error is 18% of R including idle period and 6% of R without considering the idle period. For the homogeneous Blackscholes application, which is a CPU intensive workload (while Streamcluster is a memory-intensive workload), the average tracking error is 15% of R including the idle period and 5% without considering the idle period.

We then investigate the heterogeneous workload case, where the workload con-

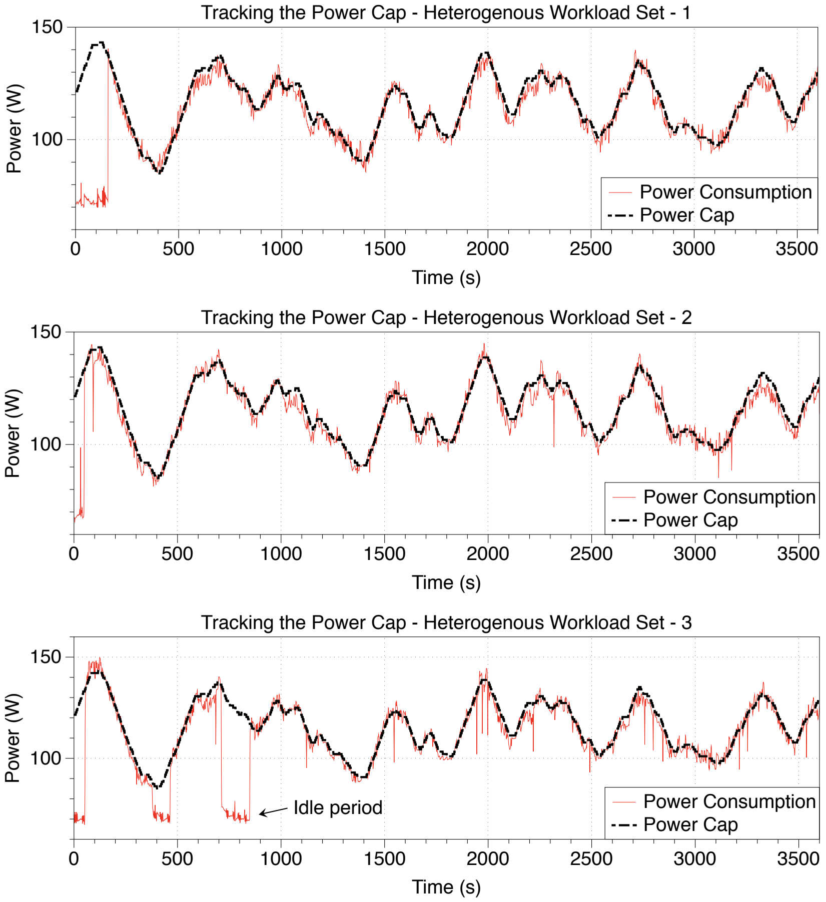


Figure 3·21: 1-hour power profile of the server running the heterogeneous workloads when we apply our proposed power capping technique.

sists of different applications. The results of the three experiments are shown in Figure 3·21. We achieve high tracking performance for all three experiments, with the average tracking errors as 15%, 10% and 16% of R including the idle period and 7%, 8% and 7% without considering the idle periods, respectively. Note that the tolerance value in the ISO requirements for the tracking error is typically 20%-30%.

All the results confirm that our runtime policy is able to track the RSR signal based power cap accurately and satisfy the tracking error tolerance in both homogeneous and heterogeneous workload cases. This result is promising as it shows that the success of the policy is not constrained by the workload type.

3.9 Energy Storage Devices (ESDs) in DR

In addition to data centers, energy storage devices (ESDs) also provide promising opportunities in DR participation. Traditionally, large scale ESDs have been deemed too expensive for widespread use in power systems; however this is beginning to change. It is interesting more than ever to understand and compare the capabilities and profits of data centers and ESDs in DR provision. Moreover, modern data centers are designed and built with ESDs as the uninterruptible power supply (UPS) units that are mainly for the purpose of bridging the time gap upon a utility failure, which in fact rarely happens. We envision that accompanied with these ESDs, data centers can provide better performance in meeting with the DR program requirements and workload QoS constraints, leading to more energy cost savings.

To this point, it is difficult to understand which storage technologies are suited for which market opportunities, and how much profit can be gained through participation. This is because different energy storage technologies have very different capabilities and constraints. In this section, we provide an overview of some potential storage technologies and define a model that enables us to study the participation of each ESD in various market opportunities.

3.9.1 Background on ESDs

We focus on five popular ESDs, namely, lead-acid (LA) batteries, lithium-ion (LI) batteries, ultra/super-capacitors (UCs), flywheels (FWs), and compressed air energy storage (CAES). In the following, we briefly highlight important characteristics of each. The interested reader can refer to prior work (Wang et al., 2012) for more information.

Lead-Acid (LA) batteries are widely used in daily life, e.g., in car batteries. They have very low self-discharge loss rates, which make them suitable for the

DR programs with long durations, e.g., hours. Additionally, they have moderate energy cost and power cost, and therefore are robust under different market scenarios. However, the key disadvantage of LA batteries is the relatively small number of charge/discharge cycles and shorter float life. LA batteries can only be used for several thousand circles.

Lithium-Ion (LI) batteries are also widely used in our daily life, and have similar characteristics to LA batteries. The key difference is that LI batteries have relatively higher costs, longer lifetimes, more cycles, and higher efficiency.

Ultra/super-Capacitors (UCs) differ dramatically from LI and LA batteries. UCs have an extremely high tolerance for frequent charging/discharging. Additionally, UCs have high efficiency and power density. However, they have a high energy cost (around $10,000/kWh) and high self-discharge rate.

Flywheels (FWs) represent a middle ground between LI/LA batteries and UCs. Like UCs, they have high efficiency and power density, but also high energy cost and a high self-discharge rate.

Compressed Air Energy Storage (CAES) has a very low energy cost and self-discharge rate. However, it has a very slow ramping time (10 min vs. 1ms in the other four ESDs). This means that it cannot adapt quickly, which limits participation of CAES in some market programs. Additionally, it has a very low energy density (large space needed) and a high power cost.

3.9.2 Modeling ESDs

There are two key components in modeling ESDs: costs (both of procurement and operation) and operation constraints (self-discharging, ramping, etc.). Operation constraints can be classified into (i) constraints imposed by the ESD technology and (ii) constraints imposed by the DR program. Constraints of type (i) are discussed here, and constraints of type (ii) are discussed in Section 3.9.3.

ESD Costs: The life span of an ESD is normally years with one-time upfront purchase/installation cost, yet participation in a DR program can span a year, a month, or even a day. In order to handle the mismatch in time granularity, we amortize the upfront cost evenly over the lifespan of the ESD. Let P_{cap} (in kW) and E_{cap} (in kWh) represent the power capacity and energy capacity of the ESD, respectively, and C^P (in \$/kW) and C^E (in \$/kWh) are the corresponding prices. Then the one-time upfront cost is[11]:

$$C^P P_{cap} + C^E E_{cap}. \tag{3.51}$$

Two factors that need to be considered to calculate the duration of use are the face-plate lifetime $T_{\max}$ and the maximal number of charge/discharge cycles L_{cyc}. Assuming the charge/discharge frequency is f, the effective duration of use is $\min\left\{T_{\max}, \frac{L_{cyc}}{f}\right\}$. Since many of the DR programs clear the credits daily, we amortize the cost of ESDs into daily prices, namely, for each type of ESD k, we define its daily power and energy capacity prices as $C_k^{P,d}$ and $C_k^{E,d}$ as:

$$C_k^{P,d} = \frac{C_k^P}{\min\left\{T_{\max}, \frac{L_{cyc}}{f_j}\right\}}, \quad C_k^{E,d} = \frac{C_k^E}{\min\left\{T_{\max}, \frac{L_{cyc}}{f_j}\right\}}, \tag{3.52}$$

where f_j is the frequency of the charge/discharge in program j. Therefore, the daily amortized cost of the ESD is:

$$C_k^{P,d} P_{cap} + C_k^{E,d} E_{cap}. \tag{3.53}$$

ESD Operation Constraints: Assume that at time t, the charge and discharge rates of an ESD are $r(t)$ and $d(t)$, respectively. We denote the total energy stored in the ESD at time t as $e(t)$, and the overall power rate from the view of the system

---

[11] Other ways of calculating the upfront cost exist (e.g., the upfront cost is selected as the maximum of the costs on power capacity and energy capacity (Wang et al., 2012)). Our method is adaptable to such calculations, e.g., an ancillary variable can be introduced to convert the selection of the maximum on power and energy capacities into two linear constraints.

level as $p(t)$. Then we have:

$$\begin{aligned} e(t) &= e(t-1) - \mu e(t-1) + r(t) - d(t),\ \forall t, \\ p(t) &= r(t)/\eta - d(t),\ \forall t, \end{aligned} \tag{3.54}$$

where μ is the self-discharge rate of the ESD, and η is the energy charging efficiency. We have $\eta < 1$, as there is always amount of loss during the ESD charge process. μ and η vary with types of ESDs. For example, UCs and FWs in general have higher efficiency than LA and LI batteries, however, they have much higher self-discharge rate.

The charge and discharge rates are also constrained by the charge/discharge capacities of the ESD, as follows:

$$0 \le r(t) \le \frac{P_{cap}}{\gamma},\ 0 \le d(t) \le P_{cap},\ \forall t, \tag{3.55}$$

where P_{cap} is the power capacity of the ESD defined before, γ is the ratio of discharge rate to charge rate. For UCs and FWs, γ is close to 1, which means they have almost same charge and discharge capacities, however for LA and LI batteries, $\gamma > 1$, representing a (much) slower recharge rate.

The amount of energy that is stored in the ESD is constrained by the ESD energy capacity E_{cap}. In addition, it is constrained by the Depth of Discharge (DoD), which helps guarantee the lifetime of the equipment:

$$(1 - DoD)E_{cap} \le e(t) \le E_{cap},\ \forall t. \tag{3.56}$$

Finally, though most ESDs are able to ramp up their discharge rate extremely fast, some ESDs, e.g., CAES, cannot. Thus, we have the discharge rate ramp up constraint:

$$d(t+1) - d(t) \le \frac{P_{cap}}{T^{ramp}},\ \forall t, \tag{3.57}$$

where T^{ramp} is the time for ESD to ramp up the discharge rate from 0 to P_{cap}.

3.9.3 Market Opportunities for ESDs

In this section, we propose detailed models of ESD participation in various electricity market programs, including RSR, contingency reserves, and peak shaving. We introduce the revenue function, $Revenue_j$ that represents the revenue received from participation in the program j, and the constraints, $Constraint_j$ that are required by the program operator. The net profit of participation equals to $Revenue_j$ minus the daily amortized cost of the ESD in Eq. (3.53). For each type of ESD k and each program j, we derive the optimal selections of ESD energy and power capacities, as well as the optimal ESD operational policy (including the amount of reserve provision, and the solution of how to dynamically charge and discharge over time, etc.) for maximizing profit. Then we evaluate applying these ESDs with today's typical capacities, and conduct sensitivity analysis of the maximal net profit on the price of reserves. Finally, we compare the benefits of these ESDs participating in each program.

RSR Provision

We first study the capabilities and profits of different ESDs in RSR provision. Based on the RSR market introduced in Section 2.2, a provider receives $\Pi^R \cdot R$ revenue for providing R (kW) amount of reserve, where Π^R is the price of the reserve. The revenue is reduced based on the tracking error of the RSR signal, i.e., $\epsilon(t) = |p(t) - Ry(t)|$, where $p(t)$ is the power rate defined in Eq. (3.54). Note that unlike the data center, ESDs do not consume power themselves, thus they do not bid the average power consumption $\bar{P}$ as what the data center does. The overall daily revenue received from RSR participation ($T = 1$ day) is:

$$Revenue_{RSR} = \Pi^R R - c \cdot \Pi^{\epsilon}(\frac{1}{T}\sum_{t=1}^{T}|p(t) - Ry(t)|), \tag{3.58}$$

where $\Pi^{\epsilon} \approx \Pi^R$, c is the penalty coefficient on the tracking error.

The provider may lose the RSR contract if the constraint on signal tracking performance is violated. We formulate this using a probabilistic constraint:

$$\sum_{t=1}^{T} \mathbb{I}_{\{|\frac{p(t)}{Ry(t)}-1|\le\rho_1\}} \ge \rho_2 T \tag{3.59}$$

where ρ_1 and ρ_2 are parameters set by the ISO. This equation shows that the probability of tracking error at each time t, (i.e., $|p(t) - Ry(t)|$) that is smaller than $\rho_1 R|y(t)|$ should be greater than or equal to ρ_2. Eq. (3.59) is equivalent to Eq. (2.2) defined in Section 2.2.

Putting Eq. (3.52) - Eq. (3.59) together, the overall optimization formulation of ESDs in RSR provision is:

$$\begin{aligned}
\max_{E_{cap},P_{cap},R,\mathbf{r},\mathbf{d},\mathbf{p},\mathbf{e}} \quad & \Pi^R R - c \cdot \Pi^{\epsilon}\frac{1}{T}\sum_{t=1}^{T}|p(t) - Ry(t)| - (C^{P,d}P_{cap} + C^{E,d}E_{cap}), \\
\textbf{s.t.} \quad & \sum_{t=1}^{T} \mathbb{I}_{\{|\frac{p(t)}{Ry(t)}-1|\le\rho_1\}} \ge \rho_2 T, \\
& e(t) = e(t-1) - \mu e(t-1) + r(t) - d(t),\ \forall t \in [1,T], \\
& p(t) = r(t)/\eta - d(t),\ \forall t \in [1,T], \\
& 0 \le r(t) \le \frac{P_{cap}}{\gamma},\ 0 \le d(t) \le P_{cap},\ \forall t \in [1,T], \\
& (1 - DoD)E_{cap} \le e(t) \le E_{cap},\ \forall t \in [0,T], \\
& d(t+1) - d(t) \le \frac{P_{cap}}{T^{ramp}},\ \forall t \in [1,T-1], \\
& P_{cap} \ge 0, E_{cap} \ge 0, R \ge 0.
\end{aligned} \tag{3.60}$$

In the formulation we use $\mathbf{r}$, $\mathbf{d}$, $\mathbf{p}$ and $\mathbf{e}$ to denote the vectors of $r(t)$, $d(t)$, $p(t)$ and $e(t)$, respectively. The objective function is to maximize the net profit of the participation, recalling that the net profit equals the revenue for providing reserves (reduced by the penalty cost on the tracking error) minus the amortized cost of ESD equipment. The constraints are imposed by both the DR program (RSR here) and the ESD technology. The decision variables of this optimization problem are:

- Power and energy capacities of the ESD, i.e., (P_{cap}, E_{cap});
- The amount of reserve provision, i.e., R;
- $\mathbf{r}$, $\mathbf{d}$, $\mathbf{p}$ and $\mathbf{e}$, which represent how the ESD is operated dynamically, i.e., the operational policy.

Case Study

To evaluate the potential value from RSR provision, we solve the above optimization formulation for the types of ESDs introduced before. We use parameters defined by prior work (Wang et al., 2012). The RSR signal $y(t)$ that we use is a real 24-hour signal from PJM (PJM, 2013). Additionally, $\rho_1 = 0.2$, $c = 1$ and $\Pi^R = \Pi^\epsilon = \$0.1/\text{kWh}$ based on today's markets (Aikema et al., 2012; PJM, 2016).

The probabilistic constraint makes Eq. (3.60) not straightforward to solve. To simplify the problem, we first study the case of $\rho_2 = 1$, in which the probabilistic constraint in Eq. (3.59) can be transformed to a deterministic constraint:

$$\left| \frac{p(t)}{Ry(t)} - 1 \right| \leq \rho_1, \forall t \in [1, T]. \tag{3.61}$$

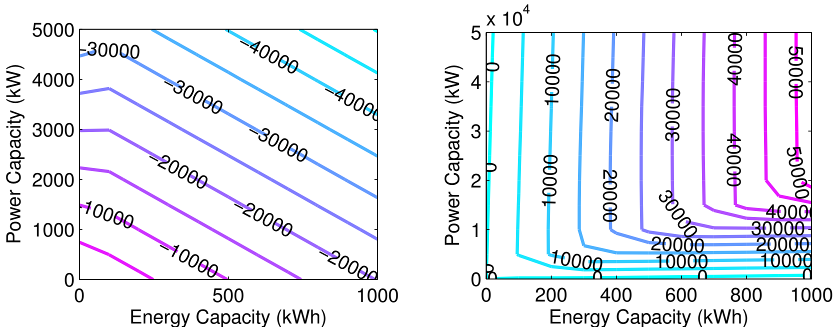
Heuristic solutions of $\rho_2 < 1$ will be discussed in Section 3.9.5. Finally, the absolute value on the tracking error in Eq. (3.60) and Eq. (3.61) leads to piecewise linear property. We simplify the piecewise linear formulation to a linear one by introducing ancillary variables $z(t)^+$ and $z(t)^-$ satisfying:

$$\begin{aligned} |p(t) - Ry(t)| &= z(t)^+ + z(t)^-, \ \forall t \in [1, T], \\ p(t) - Ry(t) &= z(t)^+ - z(t)^-, \ \forall t \in [1, T], \\ z(t)^+ \geq 0, \ z(t)^- &\geq 0, \ \forall t \in [1, T]. \end{aligned} \tag{3.62}$$

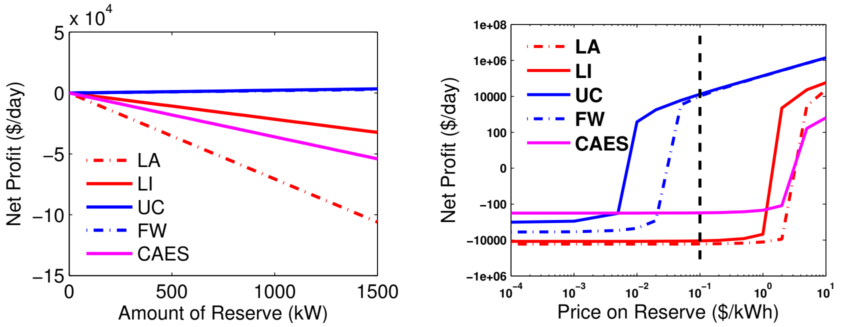
In this way, we convert Eq. (3.60) into a linear programming problem, and the optimal solution can be solved.

At the current reserve price ($\Pi^R = \$0.1/\text{kWh}$), the optimal solution of Eq. (3.60) for LA, LI batteries and CAES are all $P_{cap}^* = E_{cap}^* = R^* = 0$, which demonstrates that there is no net profit of LA, LI batteries or CAES to participate in RSR provision, i.e., the ESD cost of them is always larger than the revenue received from the provision, no matter what the power and energy capacities are used or how they are operated dynamically. On the other hand, there is no feasible optimal solution of Eq. (3.60) for UCs and FWs: the net profit keeps increasing as P_{cap}, E_{cap} and R increase, which demonstrates that the maximal net profit is large for UCs and FWs, as long as sufficiently large power and energy capacities can be offered. This highlights that the revenue earned by UCs and FWs from RSR is always larger than the amortized cost of them.

We then study the sensitivity of net profit to energy, power capacities and the amount of reserve provision. Figure 3·22(a) and Figure 3·22(b) present the optimal net profit (the negative value represents that the cost of ESD is larger than the revenue, hence the net profit is less than 0) for varying energy and power capacities (E_{cap}, P_{cap}), and for LI batteries and UCs respectively, in contour plots. LA batteries have similar results to LI batteries, and FWs are similar to UCs. From the figures, we see that for LA/LI batteries, the net profit of participating RSR is always negative, and the larger capacities of them are used, the higher cost there would be. On contrary, for UCs and FWs, a larger (E_{cap}, P_{cap}) creates larger net profit. The optimal net profit via varying amount of reserve, i.e., R, is shown in Figure 3·22(c). The net profit of LA, LI batteries and CAES is always negative and monotonously decreases along the increase of R, while the net profit of UCs and FWs is always larger than 0 and monotonously increases. Note that for all ESDs, providing larger R requires



(a) Profit of LI batteries (\$/day).   (b) Profit of UC (\$/day).



(c) Impacts of R on net profit.   (d) Impacts of price on net profit.

Figure 3·22: ESDs in RSR provision. (a) and (b) show the optimal net profit via varying energy and power capacities for LI batteries and UCs; (c) and (d) show the optimal net profit via varying amount of reserve provision, and the varying reserve price Π^R, respectively for various ESDs. The black dashed line in (d) shows the current Π^R.

larger ESD capacities.

The main factors that lead to such differences among ESDs are the characteristics of these ESDs. Since the RSR signal changes rapidly (every 4 seconds) and bi-directionally, in order to track it, RSR providers must have a large power capacity and large charge/discharge cycles. A large energy capacity, however, is not necessary, as the RSR signal has an average of zero over long time intervals. UCs and FWs perfectly match these RSR characteristics: they have extremely high tolerance for frequent charging/discharging, high efficiency and power density, and relatively low power capacity cost, whereas under the high charge/discharge frequency in RSR, the

Table 3.10: A selection of today's typical capacities of ESDs, based on space constraints.

	LA	LI	UC	FW	CAES
P_{cap} (kW)	1,000	1,000	20,000	10,000	20
E_{cap} (kWh)	250	250	250	250	250

lifetime of LA or LI batteries is shortened to less than 10 days due to the limited life cycle, which results in great cost and thus they fail to gain any net profit from RSR participation. CAES is even more limited due to the very large ramp up delay in discharge and the extremely small power density.

Next we focus on the RSR participation of different ESD technologies with today's typical capacities. In practice, the power and energy capacities of ESDs usually have upper bound limitations due to the restrictions of manufacturing techniques, unit prices and space constraints. Table 3.10 lists a selection of today's typical capacities of different types of ESDs referring to recent work (Wang et al., 2012; McCluer and Christin, 2008; Smith et al., 2008; Ghiassi-Farrokhfal et al., 2015), estimated mainly based on space constraints[12]. The power capacity of CAES is small due to its extremely small power density. The optimal net profit and the corresponding optimal R^* of these typical ESDs in RSR are listed in the 3^{rd} row of Table 3.12[13]. From the table, today's typical UCs or FWs can provide around 6MW RSR, and gain more than \$10,000 net profit a day, which are close to the power consumption and the cost of a data center with 10,000-20,000 servers. The cost of these typical UCs or FWs is around \$4 million, which can be paid back in less than one year by receiving RSR credits.

---

[12] Since we take the cost and unit prices into account in the problem formulation, we do not consider them as factors here in determining typical capacities of ESDs.

[13] All results listed in Table 3.12 are the optimization solutions of Eq. (3.60) when E_{cap} and P_{cap} are given as in Table 3.10.

Figure 3·22(d) shows the optimal net profit via varying reserve price Π^R, for different types of ESDs with their capacities fixed and given in Table 3.10. The black dashed line represents where the current market reserve price is around. From the figure, LI, LA batteries and CAES start to gain net profit (the value of the net profit is larger than 0) when the reserve price Π^R is beyond \$1/kWh.

Contingency Reserves

In ancillary markets, contingency reserves are used to respond to loss of power supplies during generation or line failures. They are typically called by the market less than once a day, and some of them are called even less than once a year. A call typically lasts from several minutes to a few hours. Reserves that are able to respond immediately are known as *spinning reserves*, whereas reserves that require more time to respond are called *non-spinning reserves.* For example, NYISO provides 10-minute spinning and 10-minute non-spinning reserves. Another types of reserves, the *operating reserves*, are also provided by NYISO, as supplements of other reserves. Operating reserves have longer reaction time but also last longer, e.g., more than 30 minutes (Aikema et al., 2012). 10-minute spinning reserves have the highest price while the price of 30-minute operating reserves is the lowest. All these prices are significantly lower than that of RSR. Overall, due to the much lower frequency of calls as well as the lower price of the reserves, the revenue received from contingency reserve provision is much lower than revenue from RSR provision.

The revenue of contingency reserve provision can be modeled as:

$$Revenue_{CR} = \Pi^{CR} R, \tag{3.63}$$

where R is the amount of contingency reserve provision and Π^{CR} is the price of the

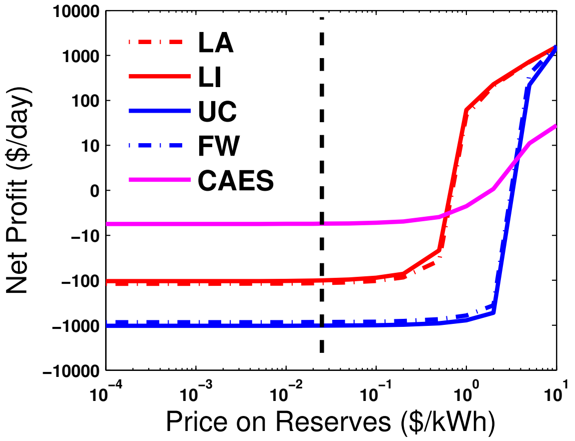


Figure 3·23: The optimal net profit via varying contingency reserve prices Π^{CR} for ESDs with today's typical capacities. The black dashed line shows the current Π^{CR}.

reserve. Unlike RSR, the contingency reserve provision is single directional with:

$$r(t) = 0, \quad d(t) = R, \ \forall t \in [T_S, T_E], \tag{3.64}$$

where $[T_S, T_E]$ is a subset of $[1, T]$, representing that only at some t during a day, an ESD is used to provide contingency reserves. For the rest of the day, the ESD is not used. When providing contingency reserves, the ESD keeps discharging at the fixed rate as the reserve value R. In order to provide the maximal amount of reserve, an ESD is charged to its full energy capacity before response, i.e.,

$$e(T_S) = E_{cap}. \tag{3.65}$$

We formulate the optimization problem for ESD in contingency reserves by putting Eq. (3.52) - (3.57) together with Eq. (3.63) - (3.65). The objective function is to maximize the net profit. The decision variables are the same as those of RSR provision.

Case Study

We focus on the 10-minute spinning reserve as an example of contingency reserves, as it is expected to have the highest revenue. $\Pi^{CR} = \$0.025/\text{kW}$ is selected for the 10-minute spinning reserve based on today's market information (Aikema et al., 2012). We assume the 10-minute spinning reserve is called once a day in our case, and $T_E - T_S = 10\text{min}$.

The optimal solution for all five ESDs in contingency reserve provision is: $P^*_{cap} = E^*_{cap} = R^* = 0$, which shows that none of five ESDs gain net profit by only providing contingency reserves at today's market reserve price, no matter what the power and energy capacities are used, and how they are operated. The larger the capacities (E_{cap}, P_{cap}) are used, the more reserves R that the ESDs can provide, however, as well as the higher the cost of ESDs would be, and the cost is always larger than the revenue from providing R.

The 4^{th} row in Table 3.12 shows results of maximal net profit of contingency reserve provision and corresponding amount of reserve provided for today's typical ESD capacities, i.e., (E_{cap}, P_{cap}) given from Table 3.10. It highlights that none of today's typical ESDs earn profit from contingency reserves at today's reserve prices. Contingency reserves are demanding in terms of energy capacity (as opposed to power capacity), though the power capacity cannot be too low either. From the table, LA and LI batteries perform better than UCs and FWs, because of their lower price on energy capacity and relatively low self-discharge rate, but are still not sufficient to be profitable. Figure 3·23 presents the optimal net profit via varying reserve prices Π^{CR} for different ESDs. LI and LA batteries start to gain profit when the price is close to \$1/kWh, whereas the critical points of CAES, UCs and FWs are around \$5-8/kWh.

Peak Shaving

The electricity bill charged monthly by utilities to large commercial and industrial power consumers, i.e., the operational expenditure (op-ex), typically consists of two parts: (i) the energy charge and (ii) the charge for the peak power during the month. The peak power is the maximum in the month of average power over each 15-30 minutes duration. The price of the peak power (i.e., the op-ex peak power price) is around \$12/kW/Month currently. The one-time cost of building power infrastructure to provide capacities to satisfy the peak power requirements, i.e., the capital expenditure (cap-ex), is around \$10-20/W on peak power based on current estimates (Wang et al., 2012). Thus, cutting peak power is an important way to reduce costs. This approach, termed peak shaving, is common and ESD provides a key method for implementation.

When participating in peak shaving, an ESD that shaves R amount of power from the peak power can gain revenue:

$$Revenue_{PS} = \Pi^{PS} R, \tag{3.66}$$

where Π^{PS} is the overall price on shaved power, i.e., the summation of the amortized capital (cap-ex) price and operational (op-ex) peak power price. The peak shaving constraints in formulation, i.e., $Constraint_{PS}$ are:

$$\begin{aligned} 0 \leq s(t) + p(t) \leq max\big(s(t)\big) - R, \ \forall t \in [1, T], \\ e(0) = e(T), \end{aligned} \tag{3.67}$$

where $s(t)$ is the power curve before peak shaving, and $max(s(t))$ is the original peak power. $p(t)$ is the power change rate from the view of system level. $s(t) + p(t)$ is the new power curve after peak shaving, and $max\big(p(t)\big) - R$ is the new peak power. $e(0) = e(T)$ represents that energy stored in the ESD is kept the same at the beginning and in the end of the time frame (in our study $T = 1$ day). We formulate the optimization problem for ESD in peak shaving by putting Eq. (3.52) - (3.57) together with Eq. (3.66) - (3.67). The objective goal is to maximize the net profit and the decision variables are the same as those in RSR provision.

Table 3.11: Optimal solutions for ESDs in peak shaving.

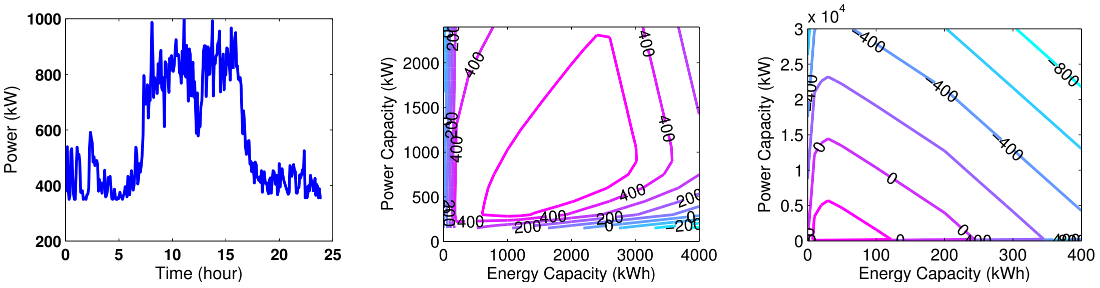
	LA	LI	UC	FW	CAES
P^*_{cap} (kW)	$1.30 * 10^3$	769.19	148.39	147.85	645.36
E^*_{cap} (kWh)	$2.15 * 10^3$	$2.40 * 10^3$	29.82	29.93	$1.83 * 10^3$
Profit (\$/day)	607.40	592.57	326.68	354.08	933.94
R^*(kW)	377.75	399.04	148.39	147.85	388.80

Case Study

We generate $s(t)$ from a real HP workload trace collected from a data center that consists of 5,000 servers. The peak power of this trace is 1MW, commonly seen in today's mid-size data center, and matches with the typical capacities of ESDs. Figure 3·24(a) is an example of $s(t)$ in a day.

Unlike the optimal solution of RSR or contingency reserve provision that is either 0 or maximal capacity allowed (i.e., no feasible optimal solution), the optimal solution of peak shaving could be in between. Table 3.11 lists the optimal solutions of different ESDs for peak shaving of the power trace $s(t)$ shown in Figure 3·24(a). All these optimal solutions lead to positive net profit. CAES has the maximal optimal net profit, though the corresponding capacities in the optimal solution are unrealistic due to its extremely small power and energy densities. LA and LI batteries have larger optimal net profit than UCs and FWs, though UCs and FWs can gain promising profit with very small capacities.

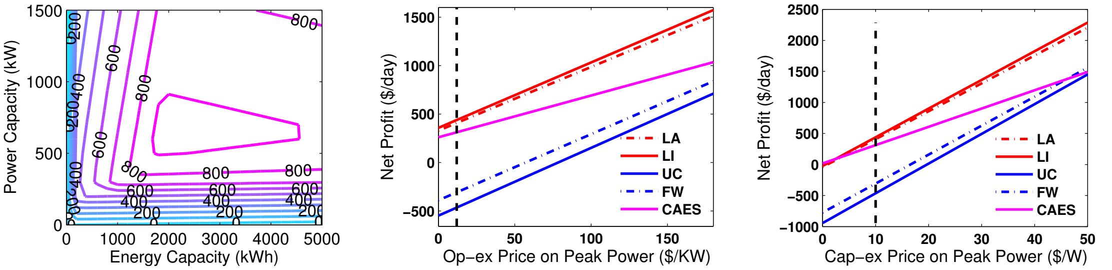
Figure 3·24(b) to 3·24(d) show the optimal net profit for varying energy and power capacities (E_{cap}, P_{cap}) in peak shaving, for LI batteries, UC and CAES, respectively. These contour plots present where the optimal solution for each ESD is located. Figure 3·24(b) also shows that LI batteries can gain profit from peak shaving in most cases, except when the power capacity is very small. In Figure 3·24(c), the profit of



(a) 1-day power trace.

(b) Profit of LI batteries ($/day).

(c) Profit of UC ($/day).



(d) Profit of CAES ($/day).

(e) Impact of op-ex price on profit.

(f) Impact of cap-ex price on profit.

Figure 3·24: ESDs in peak shaving. (a) is an example of the daily power curve before peak shaving; (b) (c) and (d) are the optimal net profit via varying energy and power capacities (E_{cap}, P_{cap}), for LI batteries, UCs and CAES respectively; (e) and (f) are the optimal net profit via varying cap-ex and op-ex peak power prices respectively for multiple ESDs. The black dash lines show where the current market prices are around.

UCs is larger than 0 only when both power and energy capacities are small, which shows that the marginal increase of the credit received from peak shaving by enlarging UC capacities is smaller than the increase in UC capacity cost. In Figure 3·24(d), CAES is always able to gain profit in peak shaving though large profit is not practical due to the limitations of power and energy densities.

Next, considering today's typical ESD capacities in peak shaving, the last row in Table 3.12 shows the optimal net profit and the corresponding optimal shaved power R^* of ESDs with typical capacities in Table 3.10, and under today's cap-ex and op-ex market prices. From the table, UCs and FWs fail to gain net profit, whereas LA, LI batteries and CAES earn net profit around \$300-400 per day.

Figure 3·24(e) and Figure 3·24(f) presents the optimal net profit of peak shaving for multiple ESDs, via varying op-ex and cap-ex peak power prices, respectively. The black dashed lines show where the current market prices are around. Note that in Figure 3·24(e), the cap-ex price is fixed at \$10/W, while in Figure 3·24(f) the op-ex price is fixed at \$12/kW/Month (both of them are current prices). Figure 3·24(e) illustrates that CAES, LI, and LA batteries gain net profit (larger than 0) under most cases including the current situation, while UCs and FWs need much higher op-ex price to gain net profit. Similar results hold for cap-ex price in Figure 3·24(f).

The peak shaving results presented here can be generalized to any scenario as long as its power trace has a similar pattern to Figure 3·24(a). This pattern is common in many scenarios (Wang et al., 2012), such as, weekday power consumption of offices, buildings and industries, power consumption of many types of data centers, e.g., data centers handling search workload (e.g., Google), communication workload (e.g., MSN), commercial and financial workload (e.g., stock exchange), etc.

3.9.4 Discussion

We provide the optimal net profit of each ESD technology across the programs in Table 3.12 for today's typical capacities and market reserve prices. From the table, LA, LI batteries and CAES gain profit from peak shaving, whereas UCs and FWs gain profit from RSR provision. None of them gain profit from contingency reserve provision, due to its low price and low calling frequency. The maximal profit earned from emerging RSR provision (by today's typical UCs or FWs) is up to 30 times of the maximal profit that can be earned from traditional peak shaving program (by LA or LI batteries), which shows that there is a great opportunity for an ESD to gain

Table 3.12: Comparing the optimal net profit of multiple types of ESDs (with E_{cap}, P_{cap} listed in Table 3.10) in participating different DR programs.

	LA		LI		UC		FW		CAES	
	Profit	R^*	Profit	R^*	Profit	R^*	Profit	R^*	Profit	R^*
RSR	-16.4k	0.17	-11.1k	0.29	13.0k	5.95	10.3k	5.94	-0.3k	0.004
CR	-0.12k	1.00	-0.10k	1.00	-1.02k	1.50	-0.85k	1.49	-0.006k	0.02
PS	0.41k	0.20	0.44k	0.20	- 0.46k	0.21	-0.31k	0.20	0.31k	0.13

---

[a] the unit of profit and R^* in table are \$/day and MW.

[b] CR: contingency reserve; PS: peak shaving.

significant profit from RSR provision in today's ancillary market. For providing RSR, UCs and FWs are the best choices due to their extremely high tolerance for frequent charging/discharging, high efficiency and power density, and relatively low power capacity cost, while LA, LI batteries and CAES are better choices for peak shaving, or contingency reserves (though are not profitable), because of their relatively lower cost on energy capacity and lower self-discharge rate.

3.9.5 Managing Participation of ESDs in RSR

Given the potential profitability of ESD participation in RSR provision, we now focus on the design of policies to enable this participation in practice. There are many challenges involved in such participation. For example, the provider is required to track an RSR signal that varies rapidly, bi-directionally, and is not known ahead of time. In addition, the revenue is deducted by tracking error, which creates a trade-off between reserve maximizing and signal tracking. In this section, we start by developing offline optimal solutions (assuming the RSR signal is known a priori), and then design practical online policies, in which the RSR signal is not known in advance.

Offline policies for RSR

In Section 3.9.3, we introduce the offline optimal solution in the case when $\rho_2 = 1$ in Eq. (3.59). $\rho_2 = 1$ simplifies the probabilistic constraint in Eq. (3.59) to a deterministic constraint in Eq. (3.61). However, normally $\rho_2 < 1$ in practice, i.e., some violations of signal tracking are tolerable, which makes the optimization problem challenging.

Policy overview

In this section, we propose three heuristic offline solutions to handle the probabilistic constraint in Eq. (3.59) when $\rho_2 < 1$. The key idea behind these solutions is to determine when the signal should be tracked within the tolerance ρ_1 (i.e., satisfying Eq. (3.61)), and when the tolerance can be violated. Three solutions are as follows:

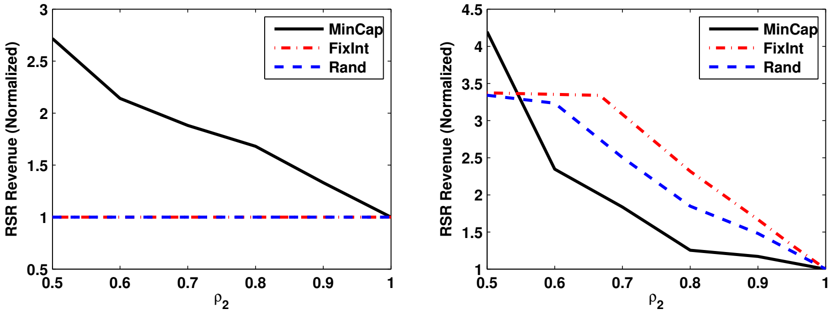
RandSelect: Randomly select $\rho_2 T$ time intervals in $[1, T]$ to satisfy Eq. (3.61).

MinCapSelect: Select $\rho_2 T$ time intervals in $[1, T]$ with smallest $|y(t)|$ to satisfy Eq. (3.61). This design is based on the fact that tracking RSR signal at the time interval t with larger $|y(t)|$ requires larger power capacity.

FixIntSelect: Equally distribute $T - \rho_2 T$ time intervals that are allowed to violate the Eq. (3.61) in $[1, T]$. This is for the purpose of enabling the policy to adjust amount of energy stored in ESDs freely (no needs to obey the tracking constraint) once a while.

Case study

Figure 3·25 shows the optimal RSR revenue solved based on Eq. (3.60) with three proposed offline methods via varying ρ_2, for LI batteries and UCs with typical capacities listed in Table 3.10, respectively. ρ_1 is fixed at 0.2, as in Section 3.9.3. Note that since we use the typical capacities in all cases, the cost of ESD is fixed. Thus, it is equivalent to make comparisons of these three methods based on either the RSR revenue, i.e., $Revenue_{RS}$ or the net profit used in the objective function of Eq. (3.60). In the figure, all the revenues are normalized by the revenue at $\rho_2 = 1$.



(a) (b)

Figure 3·25: The revenue of providing RSR via varying ρ_2, for LI batteries (in (a)) and UCs (in (b)), with three heuristic offline solutions, respectively. The revenue is normalized to the value of $\rho_2 = 1$.

From Figure 3·25(a), *MinCapSelect* always achieves largest revenue for LI batteries when ρ_2 varies. The charge/discharge capacity, i.e., the power capacity is the main bottleneck for LI batteries to offer more reserves, while *MinCapSelect* can help reduce the requirement on power capacity by only tracking small $|y(t)|$ and giving up tracking large $|y(t)|$, hence enabling LI batteries to provide additional reserves. The results for UCs, however, are different. The power capacity is no longer the bottleneck, as today's typical UCs have much stronger power capacity compared to their energy capacity. As a consequence, energy capacity turns out to be the bottleneck. In that case, *MinCapSelect* does not help, and is even worse than the random algorithm *RandSelect*. A solution that is able to utilize the limited energy capacity in a more efficient way can provide more reserves and earn higher revenue. *FixIntSelect* becomes a better solution shown in Figure 3·25(b), because it uniformly distributes time points when the tracking constraint is allowed to be violated across the whole time frame, so that the amount of energy stored in ESDs (e.g., UCs) can be adjusted periodically. Figure 3·25 also shows that the optimal revenue increases when ρ_2 decreases. Relaxing the signal tracking constraint by decreasing ρ_2 in general offers more flexibility for ESDs in RSR provision, and therefore, enables them to gain larger profit.

Online policies for RSR

Prior offline solutions are based on the fact that RSR signal is known a priori, which is, however, not for the real case in practice. RSR signal is broadcast to demand side every few seconds in real time. In this section, we propose heuristic online ESD operational policies for RSR participation, where no information on the RSR signal is required in advance. In a practical scenario, the online policies handle the following problems: given the types and capacities of the ESD (i.e., assuming the ESD has been setup), how much reserve should be provided and how the ESD should be operated so that higher revenue from RSR participation can be gained and the feasibility of the participation can be guaranteed.

Policy overview

As discussed before, *MinCapSelect* provides the highest revenue for ESDs such as LI and LA batteries in the offline solution. Hence we design the online operational policy for LI and LA batteries based on the *MinCapSelect* solution, as follows:

Initialization: we calculate two thresholds θ_0 and θ_1, based on the requirement input (ρ_1, ρ_2) from the market operator introduced before, and the historical data of RSR signal $y^H(t)$, such that:

$$\begin{aligned} \mathbf{Prob}\{|y^H(t)| \le \theta_0\} &= \rho_2, \\ \theta_1 &= (1-\rho_1)\theta_0. \end{aligned}$$

Real-time Operation: at each time t, assuming the RSR signal value is $y^r(t)$, we determine the power rate $p(t)$ by:

1. If $|y^r(t)| < \theta_1$: we set $p(t) = y^r(t)$, i.e., accurately track the signal;
2. If $\theta_0 \ge |y^r(t)| \ge \theta_1$: we set $p(t) = \theta_1 sign(y^r(t))$, i.e., cap the power rate $p(t)$ at θ_1;

3. If $|y^r(t)| > \theta_0$: we no longer track the signal, instead, we set $p(t)$ to adjust the current amount of energy stored, i.e., $e(t)$ back to a middle level $e^M = \frac{DoD \cdot E_{cap}}{2(1-\mu)}$ for future use (recall that μ is the self discharge rate);
4. Check and cap $p(t)$ and $e(t)$ based on power and energy capacity (P_{cap}, E_{cap}) constraints of the ESD.

An advanced algorithm could be updating θ_0 and θ_1 adaptively and dynamically in real time based on tracking performance feedback.

For ESDs such as UCs and FWs, the *FixIntSelect* solution offers the highest revenue from the previous study of the offline solution. Therefore, we propose the online operational policy for UCs and FWs based on the *FixIntSelect* solution, as follows:

Initialization: we calculate the intervals that adjust the stored energy in the ESD based on the input ρ_2: $T_{int} = \lceil \frac{1}{1-\rho_2} \rceil$, i.e., we adjust the stored energy every T_{int} time interval. In addition, we set $\theta_1 = 1 - \rho_1$;

Real-time Operation: at each time t, assuming the RSR signal value is $y^r(t)$, we determine the power rate $p(t)$ by:

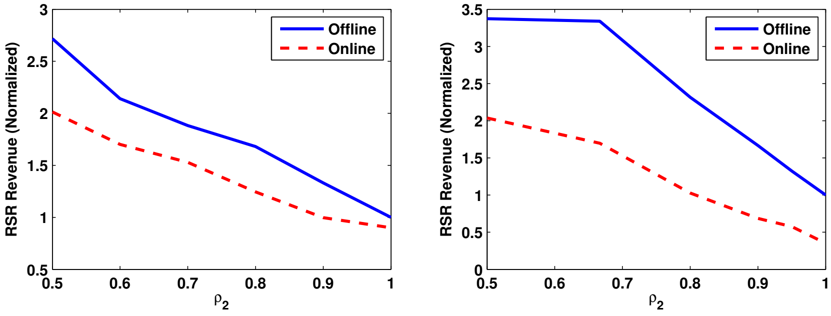
1. Every $t = T_{int}$, we set $p(t)$ to adjust the current amount of energy stored, i.e., $e(t)$ back to middle level $e^M = \frac{DoD \cdot E_{cap}}{2(1-\mu)}$;
2. For $t \neq T_{int}$, if $|y^r(t)| < \theta_1$: we set $p(t) = y^r(t)$, i.e., accurately track the signal;
3. For $t \neq T_{int}$, if $|y^r(t)| \geq \theta_1$: we set $p(t) = \theta_1 sign(y^r(t))$, i.e., cap the power rate $p(t)$ at θ_1;
4. Check and cap $p(t)$ and $e(t)$ based on power and energy capacity (P_{cap}, E_{cap}) constraints of the ESD.

Another essential issue in an online policy is the determination of the amount of reserve to provide, i.e. R_{onl}. Unlike the offline solution, in which the RSR signal is known ahead, thus an optimal R can be calculated directly from the optimization formulation, the R_{onl} for the online policies is required to be carefully estimated. We propose an approach to learn R_{onl} from historical offline solutions, as $R_{onl} = \alpha R_{min}$, where R_{min} is the minimum of the offline optimal R in the past 12 hours (the signal has been known in those hours, so offline optimal R can be calculated), α is a discount value. We use R_{min} and select α to avoid aggressive estimation of R_{onl}, and to guarantee feasibility of our policies. We select $\alpha = 90\%$ for LI batteries and $\alpha = 75\%$ for UCs, because LI batteries have more stable results, much smaller provision and are less sensitive to variations of ρ_2 than UCs shown in Section 3.9.5.

Case study

An aggressive claim of R_{onl} may lead to failure in reserve provisioning (i.e., constraints are violated) during the real-time operation, due to the limitations of ESD capacities. Hence, we first evaluate the feasibility of our online policies. We test the feasibility of our policies in the last 12 hours of a 1-day RSR signal. Each hour is a test case. In each test, we first calculate R_{onl} based on the offline optimal R in previous 12 hours as proposed, and then simulate the online policies to check whether all constraints are satisfied during the test hour. We also evaluate the policies with different ρ_2. Our results show that these safely estimated R_{onl} together with our policies satisfy all constraints and thus are feasible solutions in all test cases, for both LI batteries and UCs.

Then we compare the RSR revenue of our online policies to the offline solutions in Figure 3·26, via varying ρ_2. For offline solutions, *MinCapSelect* is selected for LI batteries, and *FixIntSelect* is selected for UCs, as they perform the best for LI batteries and UCs respectively shown in Figure 3·25, and our online policies are designed based on them. All results in Figure 3·26 are normalized to the offline solution of $\rho_2 = 1$. From the figure, the proposed online solutions still receive promising revenues, though



(a)

(b)

Figure 3·26: The revenue of providing RSR via varying ρ_2, for LI batteries (in (a)) and UCs (in (b)), respectively, with offline and online solutions. The revenue is normalized to the value of $\rho_2 = 1$ in the offline solution.

there are (as expected) noticeable gaps compared to offline solutions, due to the lack of RSR signal information, and the safe estimation of the reserve value R_{onl}. More importantly, however, the feasibility of such online policies is guaranteed with high confidence. There is the following tradeoff: an aggressive online policy may bring the revenue close to the optimal offline solution, while the real-time feasibility of such solution decreases at the same time.

3.9.6 Comparison of Data Centers with ESDs in RSR Provision

So far we have evaluated both data centers and ESDs in RSR provision. In this section, we make comparison between them. In Table 3.13, we fix the data center size and utilization at 10,000 and 50%, and study the capacities and costs that are required for different types of ESDs in order to provide the same amount of reserve as the data center could. Based on prior sections, a data center with 10,000 servers at a 50% utilization is able to provide around 550 kW RSRs to power market. Table 3.13 lists the power, energy capacities, the overall one-time upfront costs and the lifetime of different types of ESDs for the 550 kW RSR provision. From the table, the purchasing

Table 3.13: Capacities and costs of ESDs in provision of 550 kW RSR.

	LA	LI	UC	FW
P_{cap} (kW)	$2.64 \cdot 10^3$	$1.50 \cdot 10^3$	400.58	396.00
E_{cap} (kWh)	30.94	30.56	31.85	24.24
Cost (\$)	0.34 M	0.28 M	0.36 M	0.22 M
Lifetime (days)	10	26	5208	1042

cost of such ESDs is close to million dollars no matter what type of the ESD is selected. Moreover, the lifetime of common LA and LI batteries is shorter than a month in such RSR provision; in other words, these batteries are required to be replaced more than once a month, leading to tremendous costs. Overall, the comparison demonstrates that data centers can efficiently act as large-scale storage equipment in RSR provision and contribute to the power grid stabilization. Using them instead of expensive ESDs, significant amounts of monetary cost can be saved.

3.10 Summary

In this chapter, we have studied the capabilities and profits of data center participation in smart grid DR programs, especially the RSR provision. We have first modeled the data center participation in DR and RSR by introducing the detailed models of servers, clusters, workloads and their SLAs, and the overall computational units, etc. We have then proposed and evaluated three different types of runtime policies for RSR provision, i.e., the best tracking policy, the stochastic DP policy, and the EnergyQARE policy, to modulate the data center power consumption in response to the ISO request in different scenarios, by leveraging advanced power capping and budgeting techniques, various available server power states and server provisioning, and the workload arrangement. Along with runtime policies, we have also formulated optimization problems to determine the optimal energy and reserve bidding strategies in RSR provision that minimize the data center energy monetary cost, while satisfying the constraints from ISO requirements and workload SLAs. We have then evaluated the performance and the energy cost savings of RSR provision in different data center scenarios, and also made heuristic comparisons of its savings to other energy cost saving strategies. Results have demonstrated that a data center in typical scenario can achieve up to 44% monetary savings by providing RSRs, surpassing most of traditional energy saving strategies. We have then studied the real design and implementation of our proposed runtime policies and bidding strategies for data center RSR provision on a real system - a real multi-core server, which offers guidance for the future deployment of the proposed techniques onto real-life large scale data centers.

In addition to data centers, ESDs are widely studied candidates in participating DR programs. In order to compare data centers to large scale ESDs in DR, and understand how ESDs are able to assist data centers in further improving the capabilities and benefits of DR participation, we have introduced detailed models, evaluated and optimized the profit of various ESD technologies in not only legacy, but also emerging DR programs, and proposed detailed reserve value and capacity planning, as well as online ESD operational policies. Highlighted results have shown that UCs and FWs are most beneficial ESDs for RSR provision. A typical 10,000-server data center can act as million dollar level ESDs in RSR provision.

Chapter 4

Open Problems in Data Center DR

In this chapter, we introduce the open problems of data center participating in DR programs, especially the RSR provision. We propose three main directions: (1) optimize the proposed algorithms and policies of data center RSR provision with both numerical and analytical methods; (2) enable data center DR participation together with ESDs; (3) implement the DR participation on real-life data center clusters. In the following subsections, we discuss each of these directions in detail.

4.1 Optimization with Numerical and Analytical Methods

We have proposed three runtime policies for data center RSR provision. Among them, the best tracking and the EnergyQARE runtime policies handle general and practical scenarios in data centers by considering different server power states, the time delay and energy loss during the transition, power budgeting and workload arrangement, etc. All these factors coming together make it complex and challenging to seek optimal solutions. Both the best tracking and the EnergyQARE policies are heuristic solutions, though they are sufficiently effective. Furthermore, these two policies simply take the current instantaneous value of the RSR signal as the goal for tracking, and do not consider the statistical information of the signal in decision making. Understanding and leveraging the statistical information of the signal could help further improve the performance of the policies.

The stochastic DP policy, on the other hand, is an optimal stochastic policy and considers the statistical characteristics of the RSR signal. However, it assumes a simplified model of the data center that does not consider different server power states, the transitions among them, or workload heterogeneity.

Designing a generalized stochastic optimal policy by considering all the accessible information and possible control actions in data centers is an interesting open problem. For example, the current optimal stochastic DP policy can be extended by accounting for multiple server operating modes, and pursuing the characterization of optimal switching control policies, i.e., binary level control, in addition to the currently deployed continuous power consumption rate control. Workload prediction can be also integrated into the solution to further improve the performance of the policies.

In addition, we have solved the energy and reserve bidding problem optimally, i.e., $(\bar{P},\ R)$ mainly through simulation and exhaustive search. A future research direction could be investigating more efficient solutions in determining $(\bar{P},\ R)$. For example, the sensitivities of the statistics of the QoS degradation and the signal tracking error can be estimated with respect to $(\bar{P},\ R)$. Then these sensitivities can be applied to construct a more structured search for the optimal bid.

Some other control methods can be applied to search for efficient policies, e.g., model predictive control (MPC) (Camacho and Alba, 2013). Differing from the stochastic DP methods that mainly use infinite horizons, MPC uses finite (and usually short) horizons, and adaptively updates the actions based on state observations in the finite horizon. Compared with stochastic DP, MPC is more suitable for online solutions as it requires less computation and is much faster due to the usage of the short horizon. It is also shown that in some cases, the performance of MPC can be close to the performance of stochastic DP though only finite horizon is utilized.

One can use the RSR signal $y(t)$ as the reference signal in MPC, which can be predicted in short-term based on its statistics. The models, states and controls can be derived similarly as the way in stochastic DP. The constraints of the MPC are composed of the workload SLAs and the signal tracking requirements.

Another interesting direction is to search for analytical solutions for the problem. Currently we mainly use numerical methods[1], as it is highly challenging to find out analytical solutions for such a complex problem. Specifically, first, the controls are a mixture of a continuous knob (server dynamic power management) and several discrete knobs (different server power states, workload allocation, etc.). Both the service rate of each server and the number of active servers are tunable. As a result, none of widely studied queuing models (e.g., M/D/c. M/M/c, M/G/k, etc.) are directly applicable to our problem. Second, the states not only include signal $y(t)$ and its direction $d(t)$, but also include the workload QoS, which has obscure relation to the controls that cannot be simply characterized from analytical methods. Third, both the SLAs and the signal tracking constraints are formulated in the probabilistic forms, which makes it even more difficult to find an analytical solution.

Some analytical solutions are achievable if the problem is simplified with a few of assumptions. For example, if we assume that the number of servers is fixed, i.e., the only control is the service rate of each server, and we assume queue length based SLAs rather than the system time based ones, then well-studied queuing models can be applied, and analytical solutions can be possibly deduced. Analytical methods are much faster than numerical methods in general, which can be used to provide guidance in searching for slower but more effective numerical solutions.

4.2 DR Participation by Data Centers with ESDs

Today, data center infrastructures are designed associated with ESDs as the UPS, mainly for the purpose of bridging the time gap upon a utility failure. However,

---

[1] In fact, we have applied several analytical methods in our numerical solutions, e.g., we apply Little's Law in estimating the number of servers required dynamically in EnergyQARE.

these ESDs are rarely used for this purpose, and usually become active for only a few seconds when in use (Govindan et al., 2011). We believe that these mostly free ESDs can be utilized to further improve the capabilities and profits of data center in RSR provision. A future direction is to design operational policies by leveraging both data centers and ESDs together in RSR provision.

ESDs are able to assist data center in solving many issues in RSR provision. For example, from our current results of data-center-only-based RSR provision, we notice that in a number of time slots, the data center real power is much lower than the RSR signal power cap due to the lack of sufficient jobs in the system, in which scenario many servers can only be in either idle or sleep state with low and not tunable power rate. This issue can be solved by charging the additional power to ESDs. Another frequently appeared issue is when the RSR signal is low and the power budget is insufficient, either signal tracking performance or workload QoS is reduced. For this case, the stored power in ESDs can be discharged to support the workload servicing, which eliminates the performance degradation. Overall, ESDs provide additional flexibility for data center DR participation.

4.3 Real-life Implementation of DR on Data Centers

Being able to demonstrate the capabilities and benefits of DR, i.e., RSR provision on the real-life data center clusters can have significant impacts on industrial society. This dissertation has introduced the initial study on the RSR provision with a single multi-core virtualized server by CPU resource limits. The next step is to extend the technology onto a data center with multiple servers. Here we propose a simple example of the implementation.

Based on the structure proposed in Figure 3·2, we introduce master (i.e., the central controller) and slave nodes (servers in clusters). The communication proto-

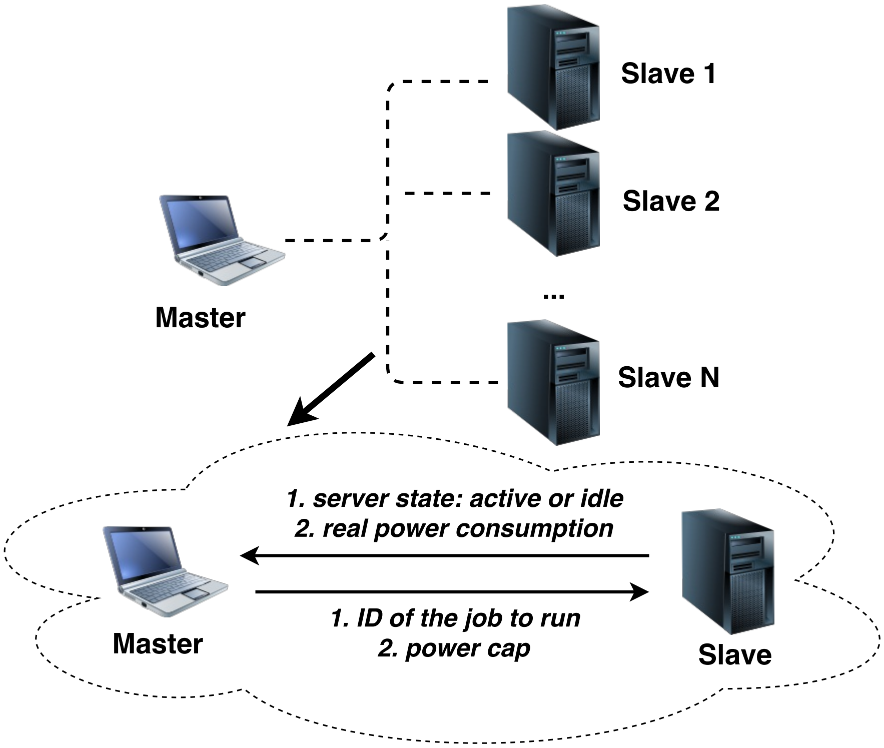


Figure 4·1: The communication between the master and slave nodes.

cols between the master node and slave nodes are designed in Figure 4·1. We use the RabbitMQ (RabbitMQ, nd), a lightweight message passing tool to send and receive information between the master and slave nodes. In implementation, all the optimization and decision making are conducted by the master node. The DR requirements and the workload arrival information are sent to the master node as well. Slave nodes are only job runners that serve jobs at the power arranged by the master node. They leverage server level power controls, such as DVFS (David et al., 2010), threads packing (Reda et al., 2012), etc., to meet with the assigned power cap. At each time interval t, the communication between the master and slave nodes consists the following two steps:

- **Step 1: slaves to master.** Slave nodes first update their state information to the master node. The information includes: (a) the server state: "active" or “idle”. Especially, if the server just finishes a job, then the finishing time is recorded, and (b) the real server power value measured by the monitoring tool. Note that slave servers that are in sleep, turned off or in transition states do not send out information. The master node directly monitors servers in these states on its side;

- **Step 2: master to slaves.** Once the master node receives the state information from slave nodes, the optimization and runtime policy engine in the master node determines the control actions, and sends the action information to slave nodes. The information includes: (a) the job ID for an idle server to run (if the server is not idle, by default it keeps running the current job). The starting time is also recorded for the newly arranged job, and (b) the power rate at which each active server should run its job.

Chapter 5

Software and System Discovery in Data Center Cloud

5.1 Overview

Cloud computing promises the delivery of on-demand computing resources as a utility that can be used as needed. This promise has led to a revolution in IT technologies causing a rapid transfer of services to the cloud (Wei et al., 2014). Regardless of whether a cloud operator uses bare metal computers, virtual machines (VMs), or containers to create computing facilities, basic questions remain the same: are these facilities free of any vulnerabilities, configured correctly, and can they avoid drifting from acceptable configuration states? New service automation and DevOps workflows have attempted to address the system drift problems by proposing the use of immutable architectures and tightly structuring software lifecycle into development, build, deployment and operations phases. However, current agile iteration principles that promote continuous development and improvement, and the fast pace of changes in underlying systems and software, counteract some of these benefits. Variability across systems in cloud environments remains a persistent problem. Therefore, discovering potential misconfiguration and vulnerability issues in a timely manner is elusive.

An effective solution to figure out system vulnerabilities and drifts is to monitor, check and analyze each change made to a system since it is booted. To understand what the system changes are about, one can dig out information from historical user or system logs. However, log data is usually too massive to be mined fast and accurately. It is also very inefficient to always keep a huge chunk of logs in storage. On the other hand, to determine if a system change includes software with known vulnerabilities, one can consult the package repository in the system and cross-check that information against, for example, National Vulnerability Database (NIST, nd). However, a vendor could issue a fix pack that fixes a known vulnerability without changing the package version. Sometimes vendors could back-port fixes into packages that have reached end of their life cycle. In both cases, a single package name links to several different versions of packages: some of them are vulnerable while others are not. Furthermore, users could install software from sources without using package managers. Simply using logs, package managers and repositories fails to discover vulnerabilities in all these scenarios.

Manually written rules that check for the existence of certain indicative features such as the existence of certain files, configuration parameters are used in addition to consulting package repositories in the system (IBM, 2012; OpenLogic, nd; OpenIOC, nd). While these rules are sufficient to detect the presence of software for license purposes, they are not capable of discriminating between a vulnerable package, and one that includes a fix for it. Furthermore, approaches based on such rules are fragile and require constant maintenance, indicating a substantial amount of manual effort. A great amount of todays software gets released multiple times a week, and most of systems change everyday. Rule-based approaches have difficulties in keeping up with the pace of software and system changes.

Alternative methodologies that build inverted indexes of file tree structures to enable keyword-based searching for software discovery are mostly useful in scenarios where users have a deep understanding of the underlying file/process structures associated with the software they are searching for and can produce specific keywords to query (Dikaiakos et al., 2012). However, as file names can be repetitive, uninformative, and misleading, the results of such systems are useful in narrowing down the search space but are not conclusive or comprehensive.

In this chapter, we introduce an automated cloud analytics solution, i.e., "discovery by example", which generates fingerprints of changes in system state, and utilizes these fingerprints in a machine learning platform to perform system change discovery and management. We first propose multiple novel feature extraction methods to generate condensed fingerprints from the comprehensive metadata associated with the system change events. Our fingerprinting methodologies mostly focus on the file system features, and tend to represent changes in system state in a compact form. They can learn the hidden context behind filenames, and represent them with vectors utilizing the file tree structure and/or file co-location information to capture the semantic relationships of files. Using these fingerprints, we build an adaptive knowledge base that enables fast comparison of system state changes with previously labeled changes. More specifically, we learn the discovery model from the knowledge base with learning algorithms and then predict the new-coming system changes by the model. We then conduct experiments mainly based on system changes caused by software installation. Typical system changes include: software installations, updates, system reconfigurations and process executions. Among them, software installation is one of the most significant factors causing system changes (Bohner, 1996). Note that, our approach, however, is applicable for discovery of system changes caused by any of the above listed factors, as the procedure of the discovery remains essentially the same and is independent of the reasons of the changes. We evaluate several machine learning algorithms as part of the proposed discovery and identification framework on our knowledge base. We show that our mechanism can be utilized for fast (in a few milliseconds or seconds) and accurate (up to 98.75%) software and system change discovery.

The chapter is organized as follows: Section 5.2 surveys the related work. Section 5.3 introduces the overall framework of the change set creation, training and discovery phases. Then we introduce the change set creation phase in detail in Section 5.4, in which we define what a change set is and how it is created. After that, we study the training phase in Section 5.5. We propose multiple fingerprinting methodologies to capture the extensive information stored in change sets in a compact form, followed by presenting various learning algorithms that we utilize for training the model. Section 5.6 first briefly introduces the system change discovery phase and the experimental methodology, then analyzes and discusses on the performance of our discovery framework. Section 5.7 summarizes this chapter and proposes the open problems.

5.2 Related Work

Standard system management and system change discovery mechanisms employed industrially today are mainly rule-based solutions that utilize large sets of manually written rules to check the existence of certain indicative properties, such as the existence of certain files. OpenIOC (OpenIOC, nd) is one such open framework that uses rules to examine registry, file content and metadata information to determine security vulnerabilities. BigFix (IBM, 2012) is a commercial offering that uses rules to scan systems and applies fixes automatically based on scan results. Rule-based approaches, however, are labor intensive as each new system or software requires a new set of rules, requires frequent edits and updates due to updates on systems and/or software packages, and requires domain expertise over a variety of systems and applications to prepare the rules, which is hard to come by.

As a complementary solution to manually written rules, a few studies investigate automated learning methods in system performance diagnosis (Bodik et al., 2010; Cohen et al., 2005; Xiong et al., 2013). These studies mainly rely on system performance metrics to detect the performance drift on either hardware or firmware layer, and mostly do not handle problems in software and system layer. EnCore (Zhang et al., 2014b) is a tool developed that learns configuration rules from a given set of sample configurations, and automatically detects software misconfigurations. Though it effectively solves types of misconfiguration problems, it does not target to general software and system changes. Registry entries and system event logs have been used in troubleshooting methods that identify problems on a given system (Yuan et al., 2006). Recently, some work studies the opportunities and challenges to interactively search across VM images at a high semantic level, and sketches the outline of an implementation by a discard-based search (Satyanarayanan et al., 2010; Huston et al., 2004). Alternative system change and software discovery methodologies based on indexing methodologies and information retrieval techniques are proposed. Minersoft (Dikaiakos et al., 2012) indexes file system information to build a keyword-based query processing system that enables searching for software existence on indexed systems. Similarly, Mirage (Ammons et al., 2011) is an image library that stores cloud images such that their file system structure is indexed in a way that enables scanning, searching and comparison of VM instances. However, indexing-based approaches require maintenance of large indexes per target VM that get constantly updated as the VM evolves. Besides, indexed file names and processes can have repetitive string representations, which can be uninformative and misleading thus results in inconclusive or incomprehensive result sets.

In this work, we propose a novel approach, the “discovery by example” method for software and system discovery in data center management. Unlike the rule-based approaches that are broadly studied, our approach (1) is fully automated requiring little to no human intervention; (2) can adapt to changes and updates by learning from the new examples and updating models; (3) significantly reduces the amount of maintenance required due to changes on instances by creating compact representations of changes occurring in system states, and (4) can provide highly accurate and comprehensive results to system change discovery queries. We also design multiple novel "fingerprints" to represent the software and systems in efficient and scalable ways, and apply a variety of machine learning algorithms for discovery.

5.3 The Framework of Discovery

Our system change discovery framework is composed of three phases: (I) change set creation, (II) training, and (III) discovery. A change set, which contains all changes that happen to the system during a system event (e.g., a software installation), is crawled and recorded in the change set creation phase. Figure 5·1 shows the change set creation flowchart. The training phase is composed of two stages: the fingerprint extraction and the model-learning. A fingerprint, a compact representation of each change set, is created in fingerprint extraction phase. In the model-learning phase, a knowledge base is first built up by change sets with known labels, and their corresponding fingerprints. The "label" here represents the name of the event that leads to the system changes. It can be a software package installation, e.g., "Apache Tomcat installation", update, e.g., "Tomcat update", or system configuration, e.g., "Tomcat configuration", etc. All fingerprints along with their labels in the knowledge base are then supplied to the learning algorithms to generate a machine learning model. Finally in the discovery phase, the learned model is utilized in the task of label prediction for new unidentified changes. Newly labeled change sets and their corresponding fingerprints are then stored into the knowledge base for future learning, which makes

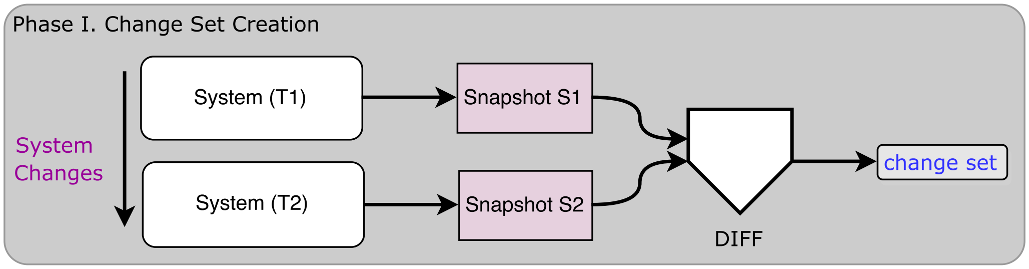


Figure 5·1: Flowchart of change set creation. Snapshots of the system are captured before and after the system change event. Then, a diff operation is calculated on these two snapshots, and the change set is generated.

the knowledge base iteratively updated. In this way, the whole discovery system is automated and requires little to no human intervention in the long-term. Manually labeled training samples are only required at the beginning of the initialization of the knowledge base. After the initialization, human operators only need to verify or clarify samples that are labeled with low confidence, which only constitute a small set of whole samples. Figure 5·2 provides an overview of the training and discovery phases.

5.4 Change Set Creation

A change set is the record of all changes that happen to the system during a system event, such as a software installation. It contains all features that are created, modified or deleted during the event, e.g., files, packages, processes and configurations. The change set creation process and an example of the change set are shown in Figure 5·1 and Figure 5·3 respectively. We create the change set by utilizing IBM's Origami service (Isci and Bala, 2014; Reimer et al., 2008). As an example to change set creation, consider the installation of a software package such as Apache Tomcat, an open source Java Servlet software. A "snapshot" S_1 of the system is taken at T_1, followed by the installation of the subject software, in this scenario Tomcat, followed

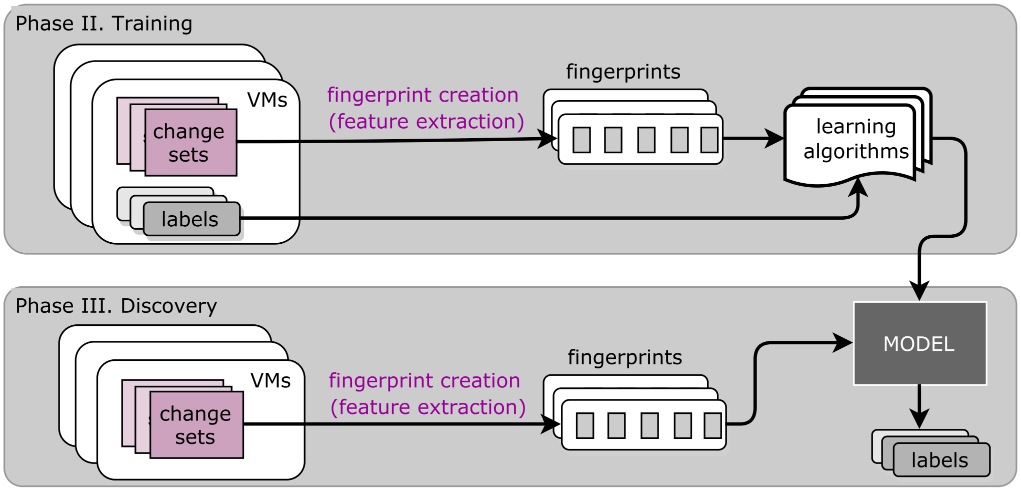


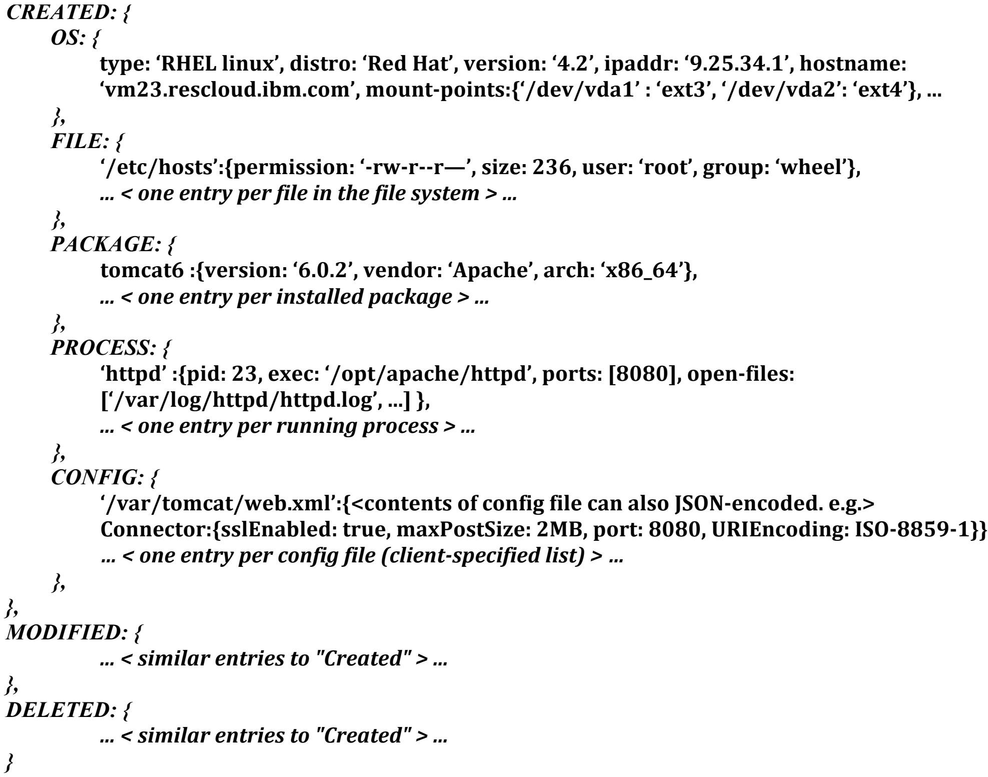
Figure 5·2: Training and discovery phases of the system change discovery framework. Labels and extracted fingerprints from change sets are input to learning algorithms to train the model in the training phase. The learned model is then used to discover and label the new-coming unidentified changes during discovery.

by a second "snapshot" S_2 of the system at T_2. The difference of two snapshots, i.e., $D = S_2 - S_1$, is a change set and we label it as "Tomcat Installation" to mark that this change set represents the system state changes observed due to an Apache Tomcat installation. More specifically:

- If a feature is in S_2 but not in S_1, then it is a *created* feature;
- If a feature is in both S_1 and S_2, but its attributes differ, then it is a *modified* feature;
- If a feature is not in S_2 but is in S_1, then it is a *deleted* feature.

Technically, a "snapshot" is taken as a text file consisted of metadata of the system, and the difference D is the output of a "text diff" applied on two snapshots.

The change set includes features from different sources. Take the Tomcat installation as an example, the file features in the change set include: (1) Tomcat server related files; (2) system and configuration files modified during installation (e.g.,



```
CREATED: {
    OS: {
        type: ‘RHEL linux’, distro: ‘Red Hat’, version: ‘4.2’, ipaddr: ‘9.25.34.1’, hostname:
        ‘vm23.rescloud.ibm.com’, mount-points:{‘/dev/vda1’ : ‘ext3’, ‘/dev/vda2’: ‘ext4’}, …
    },
    FILE: {
        ‘/etc/hosts’:{permission: ‘-rw-r--r—’, size: 236, user: ‘root’, group: ‘wheel’},
        … < one entry per file in the file system > …
    },
    PACKAGE: {
        tomcat6 :{version: ‘6.0.2’, vendor: ‘Apache’, arch: ‘x86_64’},
        … < one entry per installed package > …
    },
    PROCESS: {
        ‘httpd’ :{pid: 23, exec: ‘/opt/apache/httpd’, ports: [8080], open-files:
        [‘/var/log/httpd/httpd.log’, …] },
        … < one entry per running process > …
    },
    CONFIG: {
        ‘/var/tomcat/web.xml’:{<contents of config file can also JSON-encoded. e.g.>
        Connector:{sslEnabled: true, maxPostSize: 2MB, port: 8080, URIEncoding: ISO-8859-1}},
        … < one entry per config file (client-specified list) > …
    },
},
MODIFIED: {
        … < similar entries to "Created" > …
},
DELETED: {
        … < similar entries to "Created" > …
}
```

Figure 5·3: A sample change set. It contains all features that are created, modified or deleted during the system change event, e.g., OS, files, packages, processes and configurations.

/etc/passwd by adding Tomcat users); (3) temporary files created during installation; (4) files belonging to other software installed to satisfy dependency requirements; (5) package repository file updates, and (6) files created and modified by other activities not related to Tomcat installation, etc. Therefore, for a given Tomcat version on a specific system environment, the file features contributed by the Tomcat server installation remain the same. However, the overall file features in the change set vary from installation to installation depending on what other dependent software is installed and what other parallel activities are running during the installation process.

From a deeper investigation on the change set, we observe that a significant number of features in the *modified* catalog are from system self-updates and some backend system process activities, which are “noises” to the event that we aim to discover. In addition, when a software package is installed, the number of *deleted* features is tiny, while most of features are *created* features. Since in this work we focus on discovery of software package installations, the *created* features are the most important data to be used as indicators of the targeted event. We only use *created* features in the change set to develop the rest of technologies.

5.5 Fingerprint Creation and Learning

Training has two stages, namely fingerprint creation and learning stages. In training, fingerprints are extracted from raw change set data, stored in a knowledge base, and a discovery model is then learned from data in knowledge base. Training process and its relationship with the discovery process is shown in the upper part of Figure 5·2.

5.5.1 Fingerprint Creation

Directly utilizing the change set for discovery is not efficient, due to the fact that a change set is a complete record of raw system changes. Thus, it contains a large amount of information that is irrelevant to discovery purposes. Moreover, as the size of the change set is usually large, using the change set for discovery leads to low discovery speed and high storage costs. In addition, the change set may contain sensitive personal information from users, which can be easily exposed if directly using them for discovery.

Therefore, condensed key information is required to be extracted, either explicitly or implicitly, from change sets before they can be used to train the prediction models. The process of key information extraction is called “fingerprinting”, and the extracted key information is defined as the “fingerprint”, for each change set. In this section, we introduce multiple fingerprinting methodologies.

All fingerprinting techniques introduced here use file features in the change set, such as filenames and file paths. An example of file features can be seen in Figure 5·3. File features constitute the most significant part of change sets, and in most cases using only file features is sufficient in discovery and identifying system changes caused by software installation. It is also sufficient for other causes of system changes such as software updates and system configurations in general.

Filename Fingerprint

The most intuitive, straightforward, but storage-wise inefficient fingerprint is the *filename fingerprint*. A filename fingerprint is a list of filenames[1] of all *created* files[2] in a change set. Filename fingerprints are distinguishable because the combination of filenames of all changed files is mostly unique.

For a filename fingerprint f^n, we define its length L_{f^n}, as the number of filenames in the fingerprint. Then for any two filename fingerprints, f_1^n and f_2^n, the similarity score (α_1, α_2) between them is defined as the ratio of the number of common filenames in f_1^n and f_2^n, i.e., N_{comm}, to the length of f_1^n and f_2^n, i.e., $L_{f_1^n}$ and $L_{f_2^n}$, respectively, i.e., $\alpha_1 = N_{comm}/L_{f_1^n}$ and $\alpha_2 = N_{comm}/L_{f_2^n}$. Based on the value of (α_1, α_2), there are four different relationship between f_1^n and f_2^n:

- If $\alpha_1 \approx \alpha_2 \approx 1$, then f_1^n is similar to f_2^n;
- $\alpha_1 \approx 1$ and $\alpha_1 >> \alpha_2$, then f_1^n is contained by f_2^n;
- $\alpha_2 \approx 1$ and $\alpha_2 >> \alpha_1$, then f_2^n is contained by f_1^n;
- Neither α_1 nor α_2 is close to 1, then f_1^n and f_2^n are not similar.

---

[1] Here filenames represent the base names of files without path information.

[2] The reason of only using *created* files refers to Section 5.4.

Histogram Fingerprint

A filename fingerprint can be quite redundant and inefficient especially when a change set contains thousands of file features. Besides, typical learning algorithms can better handle with numerical features than text features. Therefore, we propose a condensed numerical representation of these filenames, i.e., the *histogram fingerprint*. The process of creating a histogram fingerprint from a filename fingerprint is as follows:

1. Convert each filename in the filename fingerprint into a numerical value using some hash function, e.g., calculating the ASCII sum of all characters that the filename contains;

2. Calculate histogram by grouping all the numerical values into a few bins, i.e., N_{bins}, and count the number of values in each bin as $C_i,\ i = 1, 2, 3...N_{bins}$;

3. Normalize histogram by: $C_i^{norm} = C_i \ / \sum_{i=1}^{N_{bins}} C_i,\ i = 1, 2, ...N_{bins}$, such that $\sum_{i=1}^{N_{bins}} C_i^{norm} = 1$. The histogram fingerprint is normalized so as to be independent of the total number of filenames in the change set. The length of histogram fingerprint is fixed at N_{bins}.

Figure 5·4 is the detailed process of the histogram fingerprint creation.

Word2vec Fingerprints

Both filename and histogram fingerprints utilize the file features as is, without trying to understand the "meaning" of the names of these files. However, it is now possible to capture the syntactic and semantic similarities and relationships between words in natural languages with no human supervision by providing significant amount of textual content to neural networks (Mikolov et al., 2013a; Mikolov et al., 2013b). Word2vec (w2v) is one such open source machine learning (neural network) toolkit

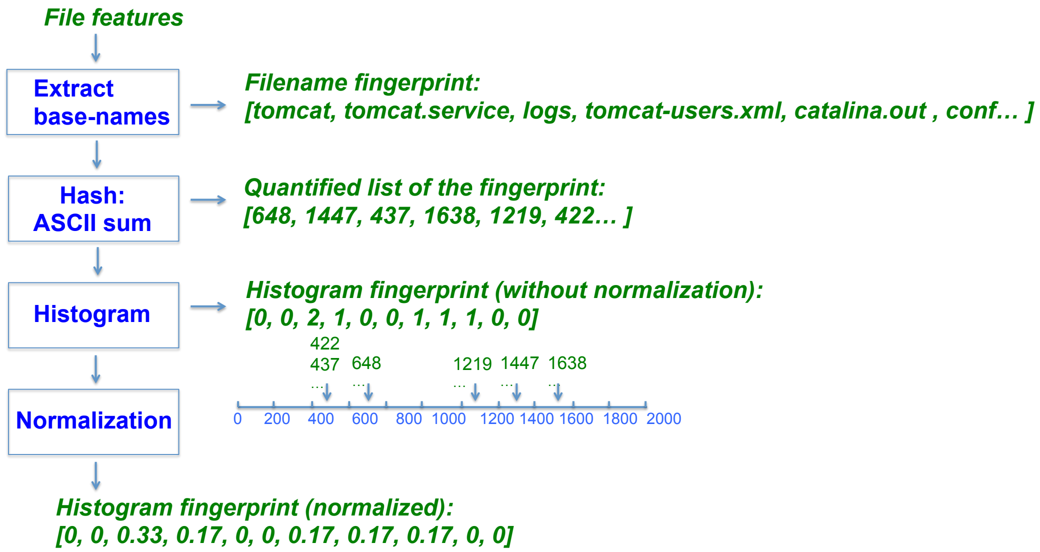


Figure 5·4: The flowchart of the histogram fingerprint generation.

developed at Google for this specific purpose (Mikolov et al., 2013a). It has been shown to successfully capture the similarities among concepts in natural languages.

We propose that w2v can also be used for gleaning the meaning behind filenames. Just as concepts that tend to appear in the same sentence in a specific order have a special relationship, we argue that filenames that appear in the same file tree branch or in the same folder (hence neighbors in locality) have a special relationship, and we propose two fingerprinting methodologies that utilize these two separate sources of information. We feed the file features and their “neighbors” - the set of files that reside in the same folder - as sentences to w2v and create a vector representation for each filename that we call “neighbor vector” of a filename. For each change set, we sum the “neighbor vectors” of the changed files in the change set by performing a simple vector addition. Then we normalize the summation vector to a unit vector to obtain a neighbor fingerprint.

Similarly, by feeding the filename of a changed file in the change set together with the folder names that are in the same file tree branch as a sentence to w2v, we create

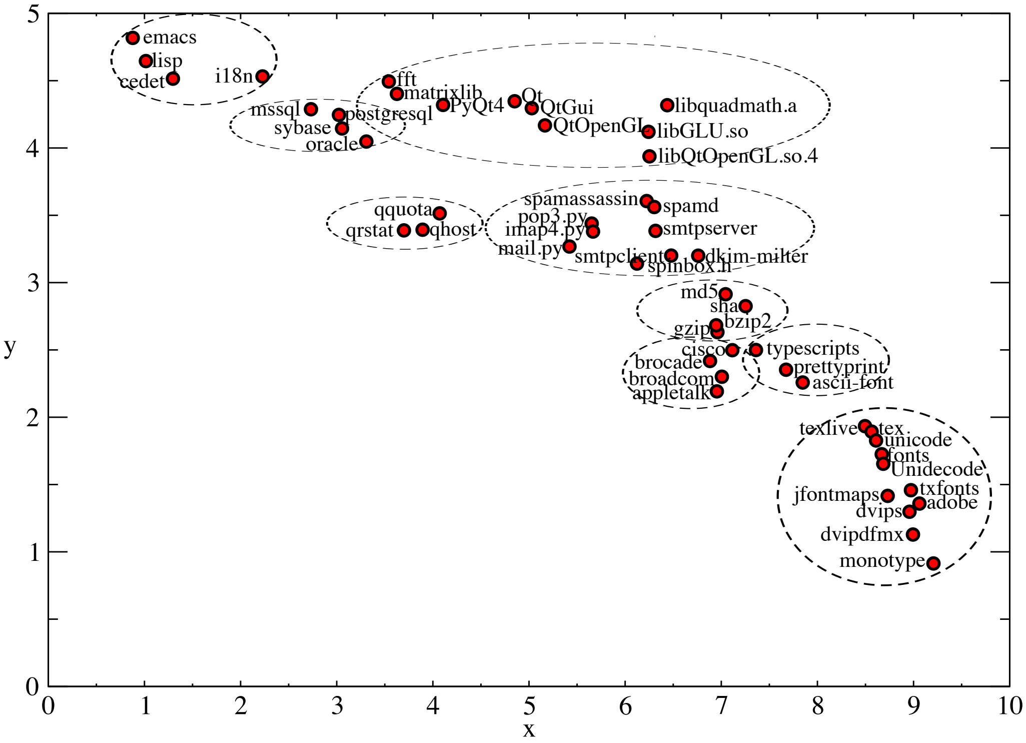


Figure 5·5: Two-dimensional (2D) vectors created by w2v for a set of filenames when file tree information is supplied to it. Created vectors retain the semantic relationship among the software objects they represent. Vector dimensions are indicated by x and y.

another vector representation for each filename, called as the "file-tree vector" of a filename. For each change set, by adding the file-tree vector representations of the changed files and then normalizing the summation vector to a unit vector, we obtain a file-tree fingerprint.

When provided with sufficient amount of folder and file tree information, we observe that w2v can easily identify the semantic relationship between files. In Figure 5·5 we display two-dimensional vectors created by w2v for a set of filenames when file tree information is supplied to it. As shown via dashed circles in the figure, even when the vector dimensions are as low as two, w2v manages to retain a sense of the semantic relationship among the software objects represented by filenames and it is even possible to roughly group the filename vectors based on these semantic relationships. As an example, it is possible to observe from the figure that the 2D vectors for Emacs - the popular Linux editor - and Lisp - the programming language used for implementing most of the editing functionality built into Emacs - are very close. Please recall that these vectors are not fingerprints themselves but they are informative inputs to the fingerprinting algorithm. Using w2v supplied vectors of changed filenames for fingerprinting enables the fingerprinting algorithm to retain a semantic sense of the installed software. When vector dimensions are increased to 200 or more, w2v starts to display much more accurate results. We should also note that w2v supplied vectors also retain a sense of relative relationship between files. As an example, when using neighbor vectors, we observe in our data that the relationship between “apache-commons-dbutils.jar” and “apache-commons-dbutils.xml” is akin to the relationship between “ivy.jar” and “ivy.xml”.

5.5.2 Learning with Fingerprints

With the condensed fingerprints, we now describe how we use these fingerprints in various learning frameworks to train models that can perform system change discovery. The set of machine learning algorithms we consider for system change discovery include nearest neighbor, logistic regression, support vector machines (SVM), decision tree and random forest. Below we briefly introduce these widely used machine learning algorithms.

Nearest neighbor (Clarke et al., 2009; Cover and Hart, 1967) is a classification technique that labels a given sample using the closest (or most similar) samples within a given previously labeled dataset. Closeness is defined by a similarity or distance function, e.g., Euclidean distance, Manhattan distance, cosine similarity, etc. A generalization of this is the k-nearest-neighbor algorithm, which utilizes the “k” closest samples. In this work, we consider the one-nearest-neighbor algorithm with the Euclidean distance. For a pair of fingerprints (f_i, f_j) introduced before, the Euclidean distance is calculated as $||f_i - f_j||$, i.e., the L_2-norm. The smaller the distance is, the more similar two fingerprints are.

Unlike other learning algorithms that have to go over a training phase to provide a learning model of coefficients, support vectors, or decision rules, the nearest neighbor algorithm requires no training. It simply keeps the set of all samples, and operates on these samples during the discovery phase to find the nearest neighbor (or k nearest neighbors) of the new-coming samples based on the given distance or similarity function, and reports the corresponding label(s) and their distances as the discovery result.

Logistic regression (Hosmer Jr and Lemeshow, 2004) is a classification algorithm that trains a coefficient vector of the feature from a training dataset by minimizing a defined cost function using programming methods. It is a generalization from linear regression by applying a logistic function. Logistic regression method can be further generalized to predict the probabilities of more than two possible outputs, i.e., the multi-class logistic regression, with applying the one-vs-all algorithm. In this work, we apply multi-class logistic regression with the L_2-regularization in our problem to avoid over-fitting. The weights on the cost of regression error and the regularization are trained through cross-validation on the training dataset.

Support Vector Machine (SVM) (Cortes and Vapnik, 1995) attempts to find an optimal set of hyper-planes in high-dimensional space that divides the samples into classes with largest margins. An SVM model is learned from training samples, which maps the samples as points in space, and divides classes by clear gaps (hyper-planes). Samples are then predicted to classes based on the side of the gap that they fall on. Samples on the margins are called support vectors. We apply one-vs-one algorithm to extend a binary SVM to a multi-class SVM, i.e., $N(N-1)/2$ classifiers are constructed if we have N classes.

SVM applies kernel functions to map the original space to a higher-dimensional space. The most widely used kernel functions are the linear kernel and the radial basis function (RBF) kernel (Hsu et al., 2003), which are both tested in our experiment. In SVM, a soft margin is typically applied, which chooses a hyper-plane that splits examples as cleanly as possible, though makes a more complex decision hype-plane. The trade-off parameter and other parameters related to different kernels are learned by cross-validation on the training dataset in our experiment.

Decision tree (Clarke et al., 2009) is a tree-like graph in which each (non-leaf) node and each branch represent a test on an attribute and the outcome of the test, respectively. Leaf nodes represent classes, into which samples are finally classified after passing through tests on all attributes. A decision tree is most commonly learned in a top-down induction method, i.e., repeatedly splitting training sets into subsets in a recursive manner based on tests of attributes until splitting no longer improves the prediction performance. Comparing with other learning algorithms, an additional benefit of a decision tree is that the decision rules that are learned from a training dataset can be usually visualized in a human-readable manner.

Random forest (Breiman, 2001) is an ensemble learning method based on decision tree. It constructs multiple decision trees in training and uses the mean or mode of the prediction of individual trees as the final output. Random forest is mainly used to solve the over-fitting issue of the decision tree.

5.6 Discovery by Examples

In the discovery phase, the models trained on the knowledge base that contains change set labels and corresponding fingerprints are utilized for performing prediction over new fingerprints extracted from unobserved change sets. More specifically, the fingerprint of a new coming unobserved change set is generated, input into the model, and the identification (i.e., the label) of the change set is returned. Discovery process and its relationship with training are displayed in the lower part of Figure 5·2.

5.6.1 Experimental Methodology

The datasets used in experimentation are generated as follows: We randomly select 160 software packages from the Linux yum repository and install these packages on two different operating systems in two different cloud environments, namely the Fedora-19 on Amazon Web Service (AWS) EC2 micro instances, and the Fedora-21 on Massachusetts Open Cloud (MOC) (Bestavros and Krieger, 2014) medium instances. Note that the approach also applies to other software systems, such as APT-like repositories, manual installation from binaries, etc. We have briefly tested them and observed similar results. In addition, the approach is independent to the location of installation, as we either only use the relative path or not use the path information at all in fingerprint design. In that way, we make sure that the same software installed in different folders can still be discovered. We record the system change set for each installation. We select software package installations as the system change trigger events because software installations are one of the most significant events that can lead to notable system changes. However, the proposed discovery technique is not limited to software installations and can be applied to a variety of system change events, such as security patches, system configurations and process execution, etc.

A change set not only includes records of changes caused by the software installation, but also contains other “background noise”, such as temporary files created automatically by the system and changes made by other user operations or irrelevant running activities in parallel, etc. Therefore, change sets consist of variations and vary from installation to installation. Even installing the same software on the same instance multiple times leads to different change sets. Moreover, dependency packages are resolved and installed during software installation. Some popular dependencies are shared by multiple software packages, and as a result, during the batch installation of 160 packages, dependencies of some later installed software packages may have already been installed during installations of prior software. Hence different orders of installations in the batch installation among these 160 software packages lead to differences in change sets. Thus, in order to capture variations in change sets, we batch install 160 software packages multiple times in random order. We install each software package 3 times on different AWS instances and 4 times on different MOC instances to create a training knowledge base. Overall, the training dataset consists of 160 software installation classes with each class containing 7 change set samples. This dataset is also used to generate the w2v dictionaries for neighbor and file-tree fingerprints.

Our testing dataset is generated as follows: we randomly select 80 software packages out of the 160 classes, and install each of them once on a separate AWS instance with Fedora-19. Then we randomly select another 80 software packages and install each of them once on a separate MOC instance with Fedora-21. The change set samples obtained from these installations are used as our discovery test cases. Therefore, our test dataset contains 160 tests in total, with 80 from AWS Fedora-19 installations and 80 from MOC Fedora-21 installations. The test dataset is generated in this way so as to capture the experimental varieties of different OSs and platforms. The accuracy of discovery is defined as the number of cases that are correctly identified among these 160 test cases, divided by 160. We test discovery accuracy of all combinations of different fingerprints methodologies and learning algorithms discussed previously.

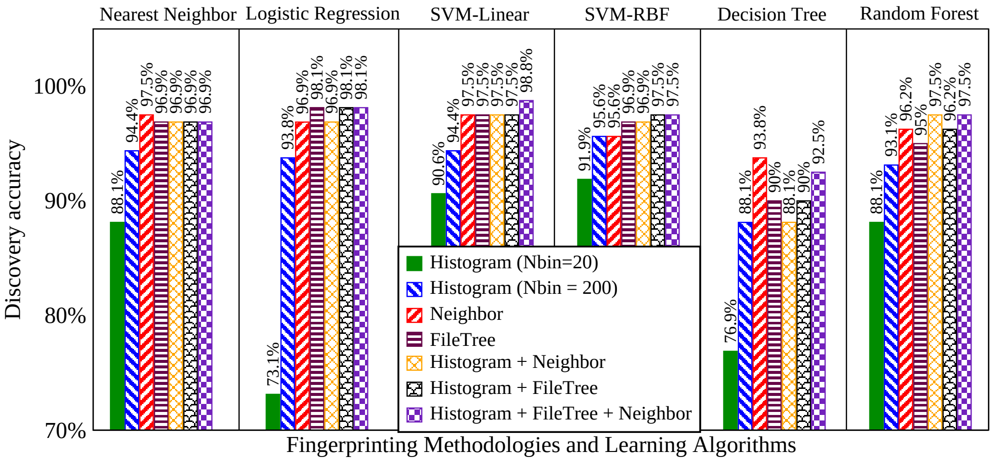


Figure 5·6: Discovery accuracy for multiple fingerprinting methodologies and learning algorithms. Results are grouped by learning algorithms.

5.6.2 Results

Figure 5·6 shows the discovery accuracy of various combinations of the fingerprinting methodologies and the learning algorithms. We test the performance of the one nearest neighbor, logistic regression with regularization, SVM with linear and RBF kernels (SVM-linear and SVM-RBF), decision tree, and random forest machine learning algorithms. In logistic regression, SVM-linear and SVM-RBF, parameters are tuned with cross-validation on the training dataset. Either one-vs-one or one-vs-all method is used in each learning algorithm for multi-class discovery, as discussed previously. Since there exist some variations in model generation in decision tree and random forest, the discovery results vary corresponding to different models. We calculate average performance of decision tree and random forest across 20 test runs.

The fingerprints in our experiment include: the histogram fingerprint with different number of bins ($N_{bins} = 20$ and $N_{bins} = 200$), the neighbor fingerprint, and the file-tree fingerprint. The lengths of both the neighbor and the file-tree fingerprints are 200. We also test the accuracy of utilizing combinations of histogram ($N_{bins} = 200$), neighbor and file-tree fingerprints as feature sets. As an example, the histogram plus neighbor fingerprint has 400 dimensions, with first 200 dimensions coming from the histogram fingerprint and the last 200 dimensions coming from the neighbor fingerprint. Similarly, the length of the histogram plus file-tree fingerprint is 400, and the length of the histogram plus file-tree and plus neighbor fingerprint is 600.

As shown in Figure 5·6, the highest discovery accuracy is as high as 98.75%, and is achieved by using SVM-linear on the combination of histogram, neighbor and file-tree fingerprints. Histogram fingerprint with 200 bins has consistently better performance than with 20 bins for all algorithms. In our experimental tests we also observe that further increasing the number of bins of the histogram to 1000 or larger counts does not increase the discovery accuracy.

We observe from Figure 5·6 that utilizing the file neighbor and file-tree information in fingerprint creation process causes notable improvements in performance. Most algorithms achieve the best performance when combinations of fingerprints are used. In some algorithms (i.e., nearest neighbor and decision tree), simply using the neighbor information leads to the highest accuracy. Involving other information such as histogram or file-tree may blur the model and predication boundary. Considering that the file-tree fingerprint depends on the paths of installation that are sometimes modified by users, neighbor information can be more reliable in broader use cases.

In addition to the discovery accuracy, the time for model training and testing is another significant aspect that should be taken into account, especially in some real-time monitoring scenarios, in which discovery results must be returned as soon as possible. From our results, all the combinations of learning algorithms and fingerprint methodologies can finish all 160 tests in less than 0.1s. We should note that this number is almost independent with the size of knowledge base in all studied algorithms except for the nearest neighbor. The test time of nearest neighbor could increase with increasing number of labeled samples in the knowledge base.

For training on a knowledge base containing 160 classes with 7 samples each, all the combinations of different fingerprinting and learning algorithms finish training in less than 20 seconds. Notice that there is no training time issue in nearest neighbor algorithm, as there is no model to be trained. In practice, a discovery system can be designed as a combination of an online training phase and an offline training phase. Algorithms that are able to train and update the model fast, though with slightly lower accuracy can be applied in the online training phase to update the prediction model frequently, while algorithms with longer training time but higher accuracy can be applied as an offline training method, to update the model less frequently with some fixed periods, e.g., once a week.

5.7 Summary and Open Problems

As cloud computing technologies continue to mature and keep gaining attractions in many industries, the demand for intelligent analytics solutions that ease the management of cloud environments increases. In this chapter we have introduced an automated cloud analytics solution, the "discovery by example" that caters to one of such demand, namely system change discovery and management. Our solution achieves efficient discovery by recording system changes in change sets, generating compact fingerprints of system state changes and utilizing these fingerprints in a machine learning platform. We have shown that with understanding the hidden context and the semantic relationships among filenames in change sets, automated, fast (in a few milliseconds or seconds) and accurate (up to 98.75%) system change discovery is achievable by our technique. The future research directions include:

Advanced Solutions in Feature Selection

Currently, we generate fingerprints only based on filenames and paths in file features. A future research direction is to study advanced feature selections. For example, our initial studies show that the size information of the files also has strong capabilities in distinguishing different software and system changes. Other features than files in the change set, such as process features, etc., may also contribute for the discovery. A more systematic method of feature selection from the raw change sets should be able to further improve the discovery accuracy and efficiency.

In addition, we apply equivalent weights on all filenames in the file features in the current work. We envision better performance if we weigh them more smartly. Some filenames, e.g., “readme”, “yum”, etc., are commonly appearing in many different change sets, which may be identified to noise with high chances. These filenames should be weighed lower than those key words in the changes, such as “tomcat”, “rabbitmq”, etc. Term frequency - inverse document frequency (tf-idf) is a potential efficient solution for this problem (Rajaraman and Ullman, 2012): each entity (i.e., a file) is weighed by a tf-idf value that increases proportionally to the number of times the word (i.e., the filename) appears in the change set, but is offset by the frequency of the word in the whole database with all the change sets. Tf-idf helps adjust for the fact that some entities appear more frequently in general, and offers key information with higher weights.

The Multi-event System Change Discovery

The proposed software and system discovery technique in this dissertation only handles the scenario that each change set perfectly captures one and only one system event (e.g., a software package installation). However, in a more realistic scenario, a change set may contain data from either partial or multiple system events, as system events can happen in parallel (e.g., several software installed together at the same time). Moreover, cloud monitoring systems usually take snapshots and change sets within a fixed period, and are not aware of whether the in-capturing system event is finished or not. In these scenarios, rather than only and perfectly having one single label for each change set, a change set should be labeled as a combination of partial or multiple labels. Therefore, a multi-event system change discovery is required. Such a problem can be solved by using the multi-label classification algorithms, e.g., applying the binary relevance method (Cherman et al., 2011) onto our existing single label learning algorithms.

Another solution is to design a two-stage multi-event discovery. In Stage 1, we identify the number of individual events included in a change set. The number of events can be estimated by various approaches. One approach is observing the histogram of file created, modified and deleted along time. Take software installation as an example. When a software package is installed, an increase in the number of files created can be observed as a "spike". We count the number of such "spikes" in the time period during which the change set is taken, and use this number as an estimation of the number of software packages installed in the change set. Note that this is only a rough estimation because on one hand, multiple events can happen simultaneously and are still overlapped in the same spike, while on the other hand, stalls are not uncommon in even a single event, which can lead to a single event with multiple spikes. Having the estimation on number of events (i.e., k) in Stage 1, then in Stage 2, on top of the binary relevance method, we further design a "confidence value based ranking" approach for discovery. Instead of directly reporting the outputs of binary classifiers, we sort the "confidence values" of all the classifiers and select the top k highest scoring labels as the final labels for the change set. Our initial results demonstrate the efficiency of this approach (Turk et al., 2016a).

Discovery as a Service

To demonstrate a real-life implementation of the work, another research direction is to architect a system that provides change discovery functionality as a service. In a typical deployment, the service would be installed on a client device and configured to perform observations of the file system on a fixed interval. Every time an observation is performed, the client would prepare a change set representing any changes made between the last observation and the current one. Afterwards, it would prepare a fingerprint using the newly generated change set and dictionaries provided by a separate server device. The prepared fingerprint is then sent off to the server for analysis, which in turn sends its prediction(s) back to the client after analysis is complete. The client finally stores the results in its log and, depending on the results received, could take appropriate actions ranging from emailing an alert to automatically quarantining the system from the network or shutting it down. Since fingerprints are highly condensed, the information transmission between client and server devices would be efficient. Moreover, as only fingerprints are transmitted to the server, which are in general not able to be reverse engineered, the sensitive personal information of cloud users is not going to be exposed on the server device and the user privacy is preserved.

Chapter 6

Conclusions

The number and size of data centers have been increasing rapidly in recent years, led by the explosive growth of the demand on world-wide Internet services and cloud computing. As a result, the data center energy and resource efficiency has started to receive significant attention due to its economical, environmental and performance impacts. In tandem, power markets operators are facing to great challenges in balancing energy supply with demand, due to the growing needs of intermittent renewable energy integration. Demand response (DR) is then introduced by the markets as an incentive to enable demand side consumers to regulate their energy consumption, to help stabilize the grids.

By investigating both the capabilities and benefits of data centers participating emerging DR programs, especially the novel regulation service reserve (RSR) provision, this dissertation has claimed that data centers provide unique opportunity to emerge as major enablers of substantial electricity integration from renewables. The participation of data centers into emerging DR, i.e., RSR provision, enables the growth of the data center in a sustainable, environmentally neutral, or even beneficial way, while also significantly reducing data center electricity costs.

In the dissertation, we have first modeled the data center participation in DR and RSR by introducing the detailed models of servers, clusters, workloads and their service level agreements (SLAs), and the overall computational units, etc. We have then specifically focused on the runtime policy design of data center in RSR provision.

While legacy DR programs such as dynamic energy pricing and peak shaving have been broadly studied recently, the RSR provision is novel to data centers. The high credits of RSRs indicate potentials in considerable savings for data centers, which however, have never been carefully investigated in literature. We have proposed and evaluated three different types of runtime policies, i.e., the best tracking policy, the stochastic dynamic program (DP) policy, and the EnergyQARE, i.e., energy and quality-of-service (QoS) aware RSR enabler, to modulate the data center power consumption in response to the ISO request in different scenarios in RSR provision, by leveraging advanced power capping and budgeting techniques, various available server power states and server commitments, as well as the workload arrangement. Along with runtime policies, we have also solved an optimization problem in data center RSR provision, to determine the optimal energy and reserve bidding strategy that minimizes the energy cost, while satisfying the constraints from ISO requirements and workload SLAs.

We have then evaluated the RSR provision performance as well as the energy monetary savings in different scenarios. To better understand the capabilities and profits of the RSR provision, we have also made heuristic comparisons of the energy cost savings from RSR to other energy saving strategies. Results have demonstrated that a typical data center can achieve up to 44% monetary savings with RSR provision, surpassing most of traditional energy saving strategies. Being an RSR provider, the data center not only receives a significant portion of monetary savings itself, but also renders massive renewable generation adoption affordable.

Moving from simulation to the practical implementation, we have also conducted initial studies on the real design and implementation of our runtime policies and bidding strategies of data center RSR provision on a real server as a data center prototype, which provides guidance for the future deployment of the proposed techniques onto real-life data centers for practical uses.

Energy storage devices (ESDs) are another promising candidates in DR participation. Data centers today are also designed associated with some ESDs. To understand how data centers can compare with large scale ESDs in RSR provision, and how the ESDs are able to be leveraged to assist data centers in further improving the capabilities and benefits of DR participation, we have modeled, optimized and evaluated the performance of different types of ESDs in participating various DR programs. Results have shown that the ultra/super - capacitors (UCs) and the flywheels (FWs) are most beneficial ESDs for RSR provision. A 10,000-server data center presents the similar capability to million-dollar level ESDs in RSR provision.

In addition to its contributions on improving data center energy efficiency, this dissertation has also proposed a novel intelligent system analytics method to address data center management efficiency and reduce the operational costs. Specifically, we have proposed a "discovery by example" approach, which leverages fingerprinting and machine learning methods to automatically discover software and system changes. We have also proposed and investigated a variety of fingerprinting designs and machine learning algorithms in discovery. Compared with the traditional rule-based discovery approach that is fragile, costly, requires specific knowledge of systems and constant maintenance by experts, our "discovery by example" approach is able to make discovery automatically, with fast speed and high accuracy, which is more suitable and efficient for today's large-scale rapidly growing data center clouds that contain great varieties of complex system changes and vulnerabilities.

References

Aalto, S., Ayesta, U., Borst, S., Misra, V., and Núñez Queija, R. (2007). Beyond processor sharing. *ACM SIGMETRICS Performance Evaluation Review*, 34(4):36–43.

Aikema, D., Simmonds, R., and Zareipour, H. (2012). Data centres in the ancillary services market. In *International Green Computing Conference (IGCC)*, pages 1–10. IEEE.

Aksanli, B., Pettis, E., and Rosing, T. (2013). Architecting efficient peak power shaving using batteries in data centers. In *Modeling, Analysis, and Simulation On Computer and Telecommunication Systems (MASCOTS)*, pages 242–253. IEEE.

Aksanli, B. and Rosing, T. (2014). Providing regulation services and managing data center peak power budgets. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–4.

Amazon (2013). Amazon EC2 service level agreement. `https://aws.amazon.com/ec2/sla`.

Amazon (2015). Amazon EC2 pricing. `http://aws.amazon.com/ec2/pricing`.

Ammons, G., Bala, V., Mummert, T., Reimer, D., and Zhang, X. (2011). Virtual machine images as structured data: The mirage image library. In *Conference on Hot Topics in Cloud Computing*, pages 22–22.

AWEA (2015). 2015 U.S. wind industry market reports. American Wind Energy Association. `http://www.awea.org/Advocacy`.

Benini, L., Bogliolo, A., and De Micheli, G. (2000). A survey of design techniques for system-level dynamic power management. *Transactions on Very Large Scale Integration (VLSI) Systems*, 8(3):299–316.

Bestavros, A. and Krieger, O. (2014). Toward an open cloud marketplace: Vision and first steps. *IEEE Internet Computing*, 18(1):72–77.

Bodik, P., Goldszmidt, M., Fox, A., Woodard, D. B., and Andersen, H. (2010). Fingerprinting the datacenter: automated classification of performance crises. In *European Conference on Computer systems*, pages 111–124. ACM.

Bohner, S. A. (1996). Impact analysis in the software change process: A year 2000 perspective. In *International Conference on Software Maintenance*, pages 42–51. IEEE.

Bohringer, C., Loschel, A., Moslener, U., and Rutherford, T. F. (2009). EU climate policy up to 2020: An economic impact assessment. *Energy Economics*, 31, Supplement 2:S295 – S305.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Burd, T. D. and Brodersen, R. W. (1995). Energy efficient CMOS microprocessor design. In *International Conference on System Sciences*, volume 1, pages 288–297. IEEE.

Camacho, E. F. and Alba, C. B. (2013). *Model predictive control.* Springer Science & Business Media.

Chase, J. S., Anderson, D. C., Thakar, P. N., Vahdat, A. M., and Doyle, R. P. (2001). Managing energy and server resources in hosting centers. *ACM SIGOPS Operating Systems Review*, 35(5):103–116.

Chen, H., Caramanis, M. C., and Coskun, A. K. (2014a). The data center as a grid load stabilizer. In *19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 105–112.

Chen, H., Caramanis, M. C., and Coskun, A. K. (2014b). Reducing the data center electricity costs through participation in smart grid programs. In *International Green Computing Conference (IGCC)*, pages 1–10. IEEE.

Chen, H., Caramanis, M. C., and Coskun, A. K. (2016a). EnergyQARE: QoS-aware data center participation in smart grid regulation service reserve provision. *submitted to ACM Transactions on Modeling and Performance Evaluation of Computing Systems (ToMPECS).*

Chen, H., Coskun, A. K., and Caramanis, M. C. (2013a). Real-time power control of data centers for providing regulation service. In *52nd Conference on Decision and Control (CDC)*, pages 4314–4321. IEEE.

Chen, H., Duri, S. S., Bala, V., Bila, N. T., Isci, C., and Coskun, A. K. (2014c). Detecting and identifying system changes in the cloud via discovery by example. In *International Conference on Big Data (Big Data)*, pages 90–99. IEEE.

Chen, H., Hankendi, C., Caramanis, M. C., and Coskun, A. K. (2013b). Dynamic server power capping for enabling data center participation in power markets. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, pages 122–129. ACM/IEEE.

Chen, H., Liu, Z., Coskun, A. K., and Wierman, A. (2015a). Optimizing energy storage participation in emerging power markets. In *6th International Green Computing Conference and Sustainable Computing Conference (IGSC)*, pages 1–6. IEEE.

Chen, H., Liu, Z., Coskun, A. K., and Wierman, A. (2015b). Optimizing energy storage participation in emerging power markets. arXiv preprint arXiv:1510.00083.

Chen, H., Turk, A., Duri, S. S., Isci, C., and Coskun, A. K. (2016b). Automated system change discovery and management in the cloud. *IBM Journal of Research and Development*, 60(2-3):2:1–2:10.

Chen, H., Zhang, B., Caramanis, M. C., and Coskun, A. K. (2015c). Data center optimal regulation service reserve provision with explicit modeling of quality of service dynamics. In *54th Conference on Decision and Control (CDC)*, pages 7207–7213. IEEE.

Cherman, E. A., Monard, M. C., and Metz, J. (2011). Multi-label problem transformation methods: a case study. *CLEI Electronic Journal*, 14(1):1–10.

Chiu, D., Stewart, C., and McManus, B. (2012). Electric grid balancing through low-cost workload migration. *ACM SIGMETRICS Performance Evaluation Review*, 40(3):48–52.

Cho, Y., Shim, J. W., Kim, S.-J., Min, S. W., and Hur, K. (2013). Enhanced frequency regulation service using hybrid energy storage system against increasing power-load variability. In *Power Energy Society General Meeting*, pages 1–5. IEEE.

Christian, B. (2011). Benchmarking modern multiprocessors. *Ph.D.Thesis. Princeton University.*

Cioara, T., Anghel, I., Bertoncini, M., Salomie, I., Arnone, D., Mammina, M., Velivassaki, T.-H., and Antal, M. (2016). Optimized flexibility management enacting data centres participation in smart demand response programs. *Future Generation Computer Systems.*

Clarke, B., Fokoue, E., and Zhang, H. H. (2009). *Principles and theory for data mining and machine learning.* Springer Science & Business Media.

Cochran, R., Hankendi, C., Coskun, A. K., and Reda, S. (2011). Pack & Cap: adaptive DVFS and thread packing under power caps. In *Proceedings of the 44th International Symposium on Microarchitecture*, pages 175–185. ACM.

Cohen, I., Zhang, S., Goldszmidt, M., Symons, J., Kelly, T., and Fox, A. (2005). Capturing, indexing, clustering, and retrieving system history. *ACM SIGOPS Operating Systems Review*, 39(5):105–118.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

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

Dayarathna, M., Wen, Y., and Fan, R. (2016). Data center energy consumption modeling: A survey. *IEEE Communications Surveys Tutorials*, 18(1):732–794.

Dhiman, G., Marchetti, G., and Rosing, T. (2009). vGreen: a system for energy efficient computing in virtualized environments. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 243–248. ACM/IEEE.

Dikaiakos, M. D., Katsifodimos, A., and Pallis, G. (2012). Minersoft: Software retrieval in grid and cloud computing infrastructures. *ACM Transactions on Internet Technology (TOIT)*, 12(1):2:1–2:34.

EIA (2014). Annual energy outlook 2014. U.S. Energy Information Administration. `http://www.eia.gov/forecasts/aeo`.

Fan, X., Weber, W.-D., and Barroso, L. A. (2007). Power provisioning for a warehouse-sized computer. *ACM SIGARCH Computer Architecture News*, 35(2):13–23.

Fooladivanda, D., Rosenberg, C., and Garg, S. (2014). An analysis of energy storage and regulation. In *International Conference on Smart Grid Communications (SmartGridComm)*, pages 91–96. IEEE.

FortCollins (n.d.). Coincident peak pricing. `www.fcgov.com/utilities/business/rates/electric/coincident-peak`.

Gandhi, A., Harchol-Balter, M., Das, R., and Lefurgy, C. (2009). Optimal power allocation in server farms. In *Proceedings of the 11th International Joint Conference on Measurement and Modeling of Computer Systems*, pages 157–168. ACM.

Gandhi, A., Harchol-Balter, M., and Kozuch, M. A. (2012). Are sleep states effective in data centers? In *International Green Computing Conference (IGCC)*, pages 1–10. IEEE.

Ghamkhari, M. and Mohsenian-Rad, H. (2012). Data centers to offer ancillary services. In *International Conference on Smart Grid Communications (SmartGridComm)*, pages 436–441. IEEE.

Ghasemi-Gol, M., Wang, Y., and Pedram, M. (2014). An optimization framework for data centers to minimize electric bill under day-ahead dynamic energy prices while providing regulation services. In *International Green Computing Conference (IGCC)*, pages 1–9. IEEE.

Ghatikar, G. (2014). Demand response opportunities and enabling technologies for data centers: Findings from field studies. Lawrence Berkeley National Laboratory. LBNL Paper LBNL-5763E. `http://escholarship.org/uc/item/7bh6n6kt`.

Ghiassi-Farrokhfal, Y., Keshav, S., and Rosenberg, C. (2015). Toward a realistic performance analysis of storage systems in smart grids. *IEEE Transactions on Smart Grid*, 6(1):402–410.

Gong, Z., Gu, X., and Wilkes, J. (2010). Press: Predictive elastic resource scaling for cloud systems. In *International Conference on Network and Service Management*, pages 9–16. IEEE.

Govindan, S., Sivasubramaniam, A., and Urgaonkar, B. (2011). Benefits and limitations of tapping into stored energy for datacenters. In *38th Annual International Symposium on Computer Architecture (ISCA)*, pages 341–351. ACM/IEEE.

Hankendi, C., Reda, S., and Coskun, A. K. (2013). vCap: Adaptive power capping for virtualized servers. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 415–420. ACM/IEEE.

Hansen, J., Knudsen, J., and Annaswamy, A. M. (2014). Demand response in smart grids: Participants, challenges, and a taxonomy. In *53rd Conference on Decision and Control (CDC)*, pages 4045–4052. IEEE.

Hosmer Jr, D. W. and Lemeshow, S. (2004). *Applied Logistic Regression*. John Wiley & Sons.

Hsu, C.-W., Chang, C.-C., Lin, C.-J., et al. (2003). A practical guide to support vector classification. `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

Hu, H., Wen, Y., Yin, L., and Qiu, L. (2016). Towards cost-efficient workload scheduling for a tango between geo-distributed data center and power grid. In *International Conference on Communications (ICC)*, pages 1–7. IEEE.

Huston, L., Sukthankar, R., Wickremesinghe, R., Satyanarayanan, M., Ganger, G. R., Riedel, E., and Ailamaki, A. (2004). Diamond: A storage architecture for early discard in interactive search. In *Conference on File and Storage Technologies (FAST)*, pages 73–86. USENIX.

Hwang, I., Kam, T., and Pedram, M. (2012). A study of the effectiveness of CPU consolidation in a virtualized multi-core server system. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 339–344. ACM/IEEE.

IBM (2012). Endpoint manager relevance language guide. `https://www.ibm.com/developerworks/community/forums/atom/download/Relevance_Guide_PDF.pdf`.

Isci, C. and Bala, V. (2014). ASPLOS 2014 tutorial - origami: Systems as data. `https://sites.google.com/site/origamisystemsasdata/asplos2014`.

Isci, C., McIntosh, S., Kephart, J., et al. (2013). Agile, efficient virtualization power management with low-latency server power states. In *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA)*, pages 96–107. ACM/IEEE.

Islam, M. A., Ren, X., Ren, S., Wierman, A., and Wang, X. (2016). A market approach for handling power emergencies in multi-tenant data center. In *International Symposium on High Performance Computer Architecture (HPCA)*, pages 432–443. IEEE.

Katz, R. H. (2009). Tech titans building boom. *IEEE Spectrum*, 46(2):40–54.

Kim, Y., Raghunathan, A., and Raghunathan, V. (2014). Design and management of hybrid electrical energy storage systems for regulation services. In *International Green Computing Conference (IGCC)*, pages 1–9. IEEE.

Kirpes, B. and Klingert, S. (2016). Evaluation process of demand response compensation models for data centers. In *Proceedings of the 5th International Workshop on Energy Efficient Data Centres*, pages 4:1–4:6. ACM.

Koomey, J. G. (2008). Worldwide electricity used in data centers. *Environmental Research Letters*, 3(3):034008.

Koomey, J. G. (2011). Growth in data center electricity use 2005 to 2010. Analytical Press. `http://www.analyticspress.com/datacenters.html`.

Kranz, B., Pike, R., and Hirst, E. (2003). Integrated electricity markets in new york. *Electricity Journal*, 16(2):54 – 65.

Kumaraswamy, K. and Cotrone, J. (2013). Evaluating the regulation market maturity for energy storage devices. *Electricity Journal*, 26(10):75–83.

Le, T. N., Liu, Z., Chen, Y., and Bash, C. (2016). Joint capacity planning and operational management for sustainable data centers and demand response. In *Proceedings of the 7th International Conference on Future Energy Systems*, pages 16:1–16:12. ACM.

Leon-Garcial, A. (2008). *Probability, Statistics, and Random Processes for Electrical Engineering*. Prentice Hall.

Li, J. and Martinez, J. F. (2006). Dynamic power-performance adaptation of parallel computation on chip multiprocessors. In *the 12th International Symposium on High-Performance Computer Architecture (HPCA)*, pages 77–87. IEEE.

Li, S., Brocanelli, M., Zhang, W., and Wang, X. (2014). Integrated power management of data centers and electric vehicles for energy and regulation market participation. *IEEE Transactions on Smart Grid*, 5(5):2283–2294.

Lin, M., Liu, Z., Wierman, A., and Andrew, L. L. (2012). Online algorithms for geographical load balancing. In *International Green Computing Conference (IGCC)*, pages 1–10. IEEE.

Liu, Z., Chen, Y., Bash, C., Wierman, A., Gmach, D., Wang, Z., Marwah, M., and Hyser, C. (2012). Renewable and cooling aware workload management for sustainable data centers. *ACM SIGMETRICS Performance Evaluation Review*, 40(1):175–186.

Liu, Z., Lin, M., Wierman, A., Low, S. H., and Andrew, L. L. (2011). Greening geographical load balancing. In *Proceedings of the SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, pages 233–244. ACM.

Liu, Z., Liu, I., Low, S., and Wierman, A. (2014). Pricing data center demand response. *ACM SIGMETRICS Performance Evaluation Review*, 42(1):111–123.

Makarov, Y. V., Loutan, C., Ma, J., and de Mello, P. (2009). Operational impacts of wind generation on california power systems. *IEEE Transactions on Power Systems*, 24(2):1039–1050.

Maly, T. and Petzold, L. R. (1996). Numerical methods and software for sensitivity analysis of differential-algebraic systems. *Applied Numerical Mathematics*, 20(1):57–79.

McCluer, S. and Christin, J.-F. (2008). Comparing data center batteries, flywheels, and ultracapacitors. *White paper*, 65.

Meisner, D., Gold, B. T., and Wenisch, T. F. (2009). PowerNap: Eliminating server idle power. *ACM SIGPLAN Notices*, 44(3):205–216.

Meisner, D., Sadler, C. M., Barroso, L. A., Weber, W.-D., and Wenisch, T. F. (2011). Power management of online data-intensive services. In *International Symposium on Computer Architecture (ISCA)*, pages 319–330. ACM/IEEE.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Miller, R. (2011). How many data centers? Emerson says 500,000. Data Center Knowledge. `http://www.datacenterknowledge.com/archives/2011/12/14/how-many-data-centers-emerson-says-500000`.

Mu'alem, A. W. and Feitelson, D. G. (2001). Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. *IEEE Transactions on Parallel and Distributed Systems*, 12(6):529–543.

Nathuji, R., Isci, C., Gorbatov, E., and Schwan, K. (2008). Providing platform heterogeneity-awareness for data center power management. *Cluster Computing*, 11(3):259–271.

Nathuji, R., Schwan, K., Somani, A., and Joshi, Y. (2009). Vpm tokens: virtual machine-aware power budgeting in datacenters. *Cluster computing*, 12(2):189–203.

NIST (n.d.). National vulnerability database. `http://nvd.nist.gov`.

NYISO (2016). Manual 2: Ancillary services manual, v4.6. `http://www.nyiso.com/public/webdocs/markets_operations/documents/Manuals_and_Guides/Manuals/Operations/ancserv.pdf`.

Ogras, U. Y., Marculescu, R., Choudhary, P., and Marculescu, D. (2007). Voltage-frequency island partitioning for gals-based networks-on-chip. In *Design Automation Conference (DAC)*, pages 110–115. IEEE.

OpenIOC (n.d.). An openIOC framework. `http://www.openioc.org`.

OpenLogic (n.d.). Oss discovery: Take inventory of your open source software. `http://ossdiscovery.sourceforge.net`.

Ott, A. L. (2003). Experience with PJM market operation, system design, and implementation. *IEEE Transactions on Power Systems*, 18(2):528–534.

Oudalov, A., Chartouni, D., and Ohler, C. (2007). Optimizing a battery energy storage system for primary frequency control. *IEEE Transactions on Power Systems*, 22(3):1259–1266.

Parekh, A. K. and Gallager, R. G. (1993). A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357.

Patel, C. D., Bash, C. E., Sharma, R., Beitelmal, M., and Friedrich, R. (2003). Smart cooling of data centers. In *International Electronic Packaging Technical Conference and Exhibition*, pages 129–137. American Society of Mechanical Engineers.

Pedram, M., Chang, N., Kim, Y., and Wang, Y. (2010). Hybrid electrical energy storage systems. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 363–368. ACM/IEEE.

PJM (2005). Integrating demand and response into the PJM ancillary service markets. White paper, PJM.

PJM (2013). Market-based regulation. `http://pjm.com/markets-and-operations/ancillary-services`.

PJM (2016). PJM manual 12: Balancing operations. `http://www.pjm.com/~/media/documents/manuals/m12.ashx`.

RabbitMQ (n.d.). `www.rabbitmq.com`.

Rajamani, K., Hanson, H., Rubio, J., Ghiasi, S., and Rawson, F. (2006). Application-aware power management. In *International Symposium on Workload Characterization*, pages 39–48. IEEE.

Rajaraman, A. and Ullman, J. D. (2012). *Mining of Massive Datasets*, volume 1. Cambridge University Press, Cambridge.

Rangan, K. K., Wei, G.-Y., and Brooks, D. (2009). Thread motion: fine-grained power management for multi-core systems. *ACM SIGARCH Computer Architecture News*, 37(3):302–313.

Rao, L., Liu, X., Ilic, M. D., and Liu, J. (2012). Distributed coordination of internet data centers under multiregional electricity markets. *Proceedings of the IEEE*, 100(1):269–282.

Reda, S., Cochran, R., and Coskun, A. K. (2012). Adaptive power capping for servers with multithreaded workloads. *IEEE Micro*, 32(5):64–75.

Reimer, D., Thomas, A., Ammons, G., Mummert, T., Alpern, B., and Bala, V. (2008). Opening black boxes: using semantic information to combat virtual machine image sprawl. In *Proceedings of the 4th SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pages 111–120. ACM.

Rivoire, S., Shah, M. A., Ranganathan, P., Kozyrakis, C., and Meza, J. (2007). Models and metrics to enable energy-efficiency optimizations. *Computer*, 40(12):39–48.

Satyanarayanan, M., Richter, W., Ammons, G., Harkes, J., and Goode, A. (2010). The case for content search of VM clouds. In *Proceedings of the 34th Annual Computer Software and Applications Conference Workshops*, pages 382–387. IEEE.

Smith, S. C., Sen, P. K., and Kroposki, B. (2008). Advancement of energy storage devices and applications in electrical power system. In *Power and Energy Society General Meeting*, pages 1–8. IEEE.

Sverdlik, Y. (2014). IDC: Amount of worlds data centers to start declining in 2017. Data Center Knowledge. `http://www.datacenterknowledge.com/archives/2014/11/11/idc-amount-of-worlds-data-centers-to-start-declining-in-2017`.

Teodorescu, R. and Torrellas, J. (2008). Variation-aware application scheduling and power management for chip multiprocessors. *ACM SIGARCH Computer Architecture News*, 36(3):363–374.

Tran, N. H., Pham, C., Ren, S., Han, Z., and Hong, C. S. (2016). Coordinated power reduction in multi-tenant colocation datacenter: An emergency demand response study. In *International Conference on Communications (ICC)*, pages 1–6. IEEE.

Turk, A., Chen, H., Byrne, A., Knollmeyer, J., Duri, S. S., Isci, C., and Coskun, A. K. (2016a). DeltaSherlock: Identifying changes in the cloud. In *submission to International Conference on Big Data (Big Data)*. IEEE.

Turk, A., Chen, H., Tuncer, O., Li, H., Li, Q., Krieger, O., and Coskun, A. K. (2016b). Seeing into a public cloud: Monitoring the massachusetts open cloud. In *USENIX Workshop on Cool Topics on Sustainable Data Centers (CoolDC)*. USENIX.

Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., and Wilkes, J. (2015). Large-scale cluster management at Google with Borg. In *Proceedings of the 10th European Conference on Computer Systems*, pages 18:1–18:17. ACM.

Villars, R., Perry, R., and Scaramella, J. (2012). Converging the datacenter infrastructure: Why, how, so, what. White paper. `https://www.emc.com/collateral/analyst-report/idc-converging-datacenter-whitepaper.pdf`.

Vu, K., Masiello, R., and Fioravanti, R. (2009). Benefits of fast-response storage devices for system regulation in ISO markets. In *Power and Energy Society General Meeting*, pages 1–8. IEEE.

Walawalkar, R., Apt, J., and Mancini, R. (2007). Economics of electric energy storage for energy arbitrage and regulation in New York. *Energy Policy*, 35(4):2558–2568.

Wang, D., Ren, C., Sivasubramaniam, A., Urgaonkar, B., and Fathy, H. (2012). Energy storage in datacenters: what, where, and how much? *ACM SIGMETRICS Performance Evaluation Review*, 40(1):187–198.

Wang, H., Huang, J., Lin, X., and Mohsenian-Rad, H. (2014). Exploring smart grid and data center interactions for electric power load balancing. *ACM SIGMETRICS Performance Evaluation Review*, 41(3):89–94.

Wang, R., Kandasamy, N., Nwankpa, C., and Kaeli, D. R. (2013a). Data centers as controllable load resources in the electricity market. In *International Conference on Distributed Computing Systems (ICDCS)*, pages 176–185. IEEE.

Wang, Y., Lin, X., Pedram, M., Park, S., and Chang, N. (2013b). Optimal control of a grid-connected hybrid electrical energy storage system for homes. In *Design, Automation and Test in Europe (DATE)*, pages 881–886.

Wei, L., Zhu, H., Cao, Z., Dong, X., Jia, W., Chen, Y., and Vasilakos, A. V. (2014). Security and privacy for storage and computation in cloud computing. *Information Sciences*, 258:371–386.

Wierman, A., Liu, Z., Liu, I., and Mohsenian-Rad, H. (2014). Opportunities and challenges for data center demand response. In *International Green Computing Conference (IGCC)*, pages 1–10. IEEE.

Xiong, P., Pu, C., Zhu, X., and Griffith, R. (2013). vPerfGuard: an automated model-driven framework for application performance diagnosis in consolidated cloud environments. In *Proceedings of the 4th International Conference on Performance Engineering*, pages 271–282. ACM.

Yang, C., Wierman, A., Shakkottai, S., and Harchol-Balter, M. (2012). Many flows asymptotics for smart scheduling policies. *IEEE Transactions on Automatic Control*, 57(2):376–391.

Yuan, C., Lao, N., Wen, J.-R., Li, J., Zhang, Z., Wang, Y.-M., and Ma, W.-Y. (2006). Automated known problem diagnosis with event traces. *ACM SIGOPS Operating Systems Review*, 40(4):375–388.

Zhan, X. and Reda, S. (2013). Techniques for energy-efficient power budgeting in data centers. In *Proceedings of the 50th Annual Design Automation Conference (DAC)*, pages 1–7. ACM/IEEE.

Zhang, B., Caramanis, M. C., and Baillieul, J. (2014a). Optimal price-controlled demand response with explicit modeling of consumer preference dynamics. In *53rd Conference on Decision and Control (CDC)*, pages 2481–2486. IEEE.

Zhang, J., Renganarayana, L., Zhang, X., Ge, N., Bala, V., Xu, T., and Zhou, Y. (2014b). EnCore: Exploiting system environment and correlation information for misconfiguration detection. *ACM SIGPLAN Notices*, 49(4):687–700.

Zhang, L., Ren, S., Wu, C., and Li, Z. (2015). A truthful incentive mechanism for emergency demand response in colocation data centers. In *Conference on Computer Communications (INFOCOM)*, pages 2632–2640. IEEE.

Zhao, C., Topcu, U., and Low, S. H. (2012). Frequency-based load control in power systems. In *American Control Conference (ACC)*, pages 4423–4430. IEEE.

Zhu, D., Wang, Y., Yue, S., Xie, Q., Pedram, M., and Chang, N. (2013). Maximizing return on investment of a grid-connected hybrid electrical energy storage system. In *19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 638–643. IEEE.

CURRICULUM VITAE

HAO CHEN

Boston University, Electrical and Computer Engineering Department
8 Saint Mary's Street, Boston, MA 02215 USA
Phone: 1-857-205-9966, Email: haoc@bu.edu
Web: http://people.bu.edu/haoc

Education

Ph.D., Boston University, Boston, MA, 09/2016
Electrical and Computer Engineering Department
Advisors: Ayse K. Coskun and Michael C. Caramanis
Dissertation: Improving Data Center Efficiency Through Smart Grid Integration and Intelligent Analytics
GPA: 3.84/4.00

B.S., Zhejiang University, Hangzhou, China, 07/2010
Information and Communication Engineering Department
GPA: 3.91/4.00, Honors Degree

Research Interests

Data center energy and resource management, green computing, smart grid demand response, operations research;
Intelligent analytics in cloud management, cloud efficiency and reliability, big data, data mining and machine learning, software.

Professional Experience

IBM T.J. Watson Research, Yorktown Heights, NY, 05/2015 - 09/2015
Cloud Research Intern, *Supervisors*: Dr. Canturk Isci, Dr. Sastry S. Duri
Investigated software, system and vulnerability discovery in cloud with fingerprinting and machine learning techniques. Improved IBM Vulnerability Advisor.

IBM T.J. Watson Research, Yorktown Heights, NY, 06/2013 - 09/2013
Software Research Intern, *Supervisors*: Dr. Vasanth Bala, Dr. Sastry S. Duri
Investigated a novel "discovery by example" technique for software, system and vulnerability discovery in cloud with image processing techniques.

Orbeus Inc. (acquired by Amazon), Excelerate Labs, Chicago, IL, 06/2012 - 08/2012
Software Engineer and Research Scientist Intern
Developed the backend infrastructure and the multi-threaded client-server parallel computing system of an online API platform and several mobile applications. Designed the HDFS and a large-scale database (Hbase + Mysql + Cassandra) for data management. Researched on face recognition and scene understanding algorithms.

Boston University Chinese Students and Scholars Association (BUCSSA), Boston, MA, 05/2011 - 05/2012
Vice President
Co-led a team to serve and facilitate communication among the large Chinese community not only on Boston University campus but also in Great Boston area.

Boston University College of Engineering, Boston, MA, 09/2010 - 05/2012
Graduate Teaching Fellow
Classes Taught: Computer Network, Applied Algorithms for Engineers, Senior Project Design I and II.

China Unicom, Hangzhou, China, 03/2010 - 07/2010
Software and Network Engineer Intern
Studied problem diagnosis and maintenance of both fixed and mobile communication networks.

Research Experience

Performance and Energy-Aware Computing (PEAC) Lab, Boston University, Boston, MA, 09/2012 - 07/2016
Research Assistant, *Supervisors*: Prof. Ayse K. Coskun, Prof. Michael Caramanis

Project 1: Data Center Smart Grid Integration

- Studied capabilities and benefits of data centers in participating smart grid demand response and green computing programs, leveraging server power management, workload allocation and control, and energy storage devices;
- Derived mathematical models, and solved the problems with stochastic and numerical methods including queuing theory, optimizations, stochastic dynamic programming and the Monte Carlo method;
- Implemented the smart grid demand response participation on the real life cluster - the Massachusetts Open Cloud (MOC).

Project 2: Cloud Management Through Intelligent Analytics

- Studied a “discovery by example” approach for software, system and vulnerability discovery in the cloud, with multiple data mining, fingerprinting and machine learning methods;
- Developed the techniques as a cloud service, collected data and tested on multiple platforms including AWS, GCE, IBM Cloud and MOC, with both virtual machines and Docker containers.

Information and Data Science Group, Boston University, Boston, MA, 06/2011 - 12/2011
Research Assistant, *Supervisors*: Prof. Janusz Konrad, Prof. Prakash Ishwar
Designed video surveillance sensor network to detect abnormal shakes, using image and video processing. Derived the earthquake spread and prediction models with Kalman Filters and data fusion techniques. Developed software to automatically visualize the earthquake information on Google Earth.

Publications and Patents

Journals:

- H. Chen, A. Turk, S. S. Duri, C. Isci, and A. K. Coskun. **Automated System Change Discovery and Management in the Cloud**, *IBM Journal of Research and Development*, 2015.
- H. Chen, M. C. Caramanis, and A. K. Coskun. **EnergyQARE: QoS-Aware Data Center Participation in Smart Grid Regulation Service Reserve Provision**, *submitted to ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, 2016.

Conferences:

- A. Turk, H. Chen, A. Byrne, J. Knollmeyer, S. S. Duri, C. Isci, and A. K. Coskun. **DeltaSherlock: Identifying Changes in the Cloud**, *submitted to IEEE International Conference on Big Data*, 2016.
- A. Turk, H. Chen, O. Tuncer, H. Li, Q. Li, O. Krieger and A. K. Coskun. **Seeing into a Public Cloud: Monitoring the Massachusetts Open Cloud**, *USENIX Workshop on Cool Topics on Sustainable Data Centers (CoolDC)*, 2016.
- H. Chen, B. Zhang, M. C. Caramanis and A. K. Coskun. **Data Center Optimal Regulation Service Reserve Provision with Explicit Modeling of Quality of Service Dynamics**, *IEEE Conference on Decision and Control (CDC)*, 2015. **(Speaker)**

- H. Chen, Z. Liu, A. K. Coskun and A. Wierman. **Optimizing Energy Storage Participation in Emerging Power Markets**, *IEEE International Green and Sustainable Computing Conference (IGSC)*, 2015.
- H. Chen, S. S. Duri, V. Bala, N. T. Bila, C. Isci and A. K. Coskun. **Detecting and Identifying System Changes in the Cloud via Discovery by Example**, *IEEE International Conference on Big Data*, 2014. **(Speaker)**
- H. Chen, M. C. Caramanis and A. K. Coskun. **Reducing the Data Center Electricity Costs Through Participation in Smart Grid Programs**, *IEEE International Green Computing Conference (IGCC)*, 2014.
- H. Chen, M. C. Caramanis and A. K. Coskun. **The Data Center as a Grid Load Stabilizer**, *IEEE Asia and South Pacific Design Automation Conference (ASPDAC)*, 2014.
- H. Chen, C. Hankendi, M. C. Caramanis and A. K. Coskun. **Dynamic Server Power Capping for Enabling Data Center Participation in Power Markets**, *IEEE/ACM International Conference on Computer-aided Design (ICCAD)*, 2013. **(Speaker)**
- H. Chen, A. K. Coskun and M. C. Caramanis. **Real-Time Power Control of Data Centers for Providing Regulation Service**, *IEEE Conference on Decision and Control (CDC)*, 2013. **(Speaker)**

Patents:

- V. Bala, H. Chen and S. S. Duri. **Creating Knowledge Base of Similar Systems from Plurality of Systems**, US20160088120, Mar. 2016.
- Y. Li, T. Liu and H. Chen. **System, Method and Apparatus for Facial Recognition**, US9275269, Mar. 2016.
- Y. Li, T. Liu and H. Chen. **System, Method and Apparatus for Scene Recognition**, US9129148, Sept. 2015.
- Y. Li, T. Liu and H. Chen. **System and Method for Automatically Generating Albums**, filed, Nov. 2013.