Carnegie Mellon University

CARNEGIE INSTITUTE OF TECHNOLOGY

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF Doctor of Philosophy

TITLE **A Learning-Based Framework Incorporating Domain Knowledge for Performance Modeling**

PRESENTED BY **Da-Cheng Juan**

ACCEPTED BY THE DEPARTMENT OF

Electrical and Computer Engineering

___Diana Marculescu_____ ____6/1/14_____
ADVISOR, MAJOR PROFESSOR DATE

_____Jelena Kovacevic_____ _6/1/14_____
DEPARTMENT HEAD DATE

APPROVED BY THE COLLEGE COUNCIL

___Vijayakumar Bhagavatula_____ ___6/1/14_____
DEAN DATE

A Learning-Based Framework Incorporating Domain Knowledge for Performance Modeling

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Da-Cheng Juan

M.S., Computer Science, National Tsing Hua University
B.S., Computer Science, National Tsing Hua University

Carnegie Mellon University
Pittsburgh, PA

June, 2014

Abstract

Energy efficiency has become the critical factor of computing performance in platforms from embedded devices, portable electronics to servers in datacenters. Power density and peak power demands increase in each generation of microprocessors, which directly lead to a higher operating temperature that exceeds the cooling capability available on current multi-core systems. These physical constraints seriously hinder the development of the next-generation computing platform. In this thesis, we propose ***Gray-Box computing****, the methodology of a learning-based framework that incorporates the prior domain knowledge to quantitatively model every aspect of a multi-core system, including performance, power consumption and operating temperature. Experimental results show that the learned model achieves more than 96% accuracy, compared to actual industrial measurements or full-system simulations. By exploiting the learned model, the proposed Gray-Box computing has enabled a wide variety of applications − from simulation speedup to multi-constrained optimization with respect to performance, energy efficiency and reliability for a multi-core system. Gray-Box computing has also been extended to model the performance−specifically, the job inter-arrivals−of a datacenter, which is in the scale of tens of thousands of cores. Experimental results are poised to demonstrate the strength of Gray-Box computing. Future work will focus on applying Gray-Box computing to model the usage dynamics of datacenters.*

Acknowledgements

The past five years I have spent at Carnegie Mellon have been a wonderful feast of knowledge−a feast that I could enjoy for the rest of my life−because of these brilliant, sincere and wise colleagues, collaborators and friends.

First and foremost, I would like to thank my advisor, Prof. Diana Marculescu. I couldn't find a better advisor than Diana anywhere around the world. During the past five years, I found myself learning something new and meaningful each time when interacting with her. Her patience guides me, step by step, through the winding path of seeking for truth and knowledge, and her wisdom unveils the true joy of the Ph.D. journey for me.

I am very grateful to Prof. Christos Faloutsos. I have been greatly inspired by his insight of conducting research and his enthusiasm of teaching. Every discussion with him is joyful. One day, I aspire to come close to the role model he has set to me as a wonderful teacher, a brilliant researcher, and ultimately, a supportive mentor.

I would like to thank Prof. Xin Li and Dr. Jinpyo Park for being my collaborators and committee members. They have provided valuable comments for further polishing this thesis.

I would like to thank all my colleagues and collaborators: Mr. Huan-Kai Peng, Prof. Siddharth Garg, Dr. Lei Li, Dr. Zhiliang Qian, Prof. Paul Bogdan, Prof. Chi-Ying Tsui, Prof. Radu Marculescu, Mr. Huapeng Zhou, Dr. Yi-Lin Chuang, Prof. Yao-Wen Chang, Dr. Raj Yavatkar, Dr. Denver Dash, Dr. Michael Kishinevsky, Prof. Umit Ogras for their constructive and professional suggestions.

I would also like to thank all my friends for their numerous discussions, useful information, and encouragement:

Prof. Kai-Chiang Wu, Prof. Natasa Miskov-Zivanov, Mr. Guangshuo Liu, Mr. Ermao Cai, Ms. Jiajia Jiao, from EnyAC;

Dr. Wangyang Zhang, Dr. Jinyin Zhang, Dr. Andrea Bartolini, Mr. Alexey Tumanov, Dr. Cheng-Yuan Wen, Mr. Jiun-Ren Lin, Mr. Kuan-Chieh Chen, Mr. Cory Bevilacqua, Mr. Radu David, Dr. Qiuling Zhu, Dr. Tam Wing Chiu, Dr. Evangelos Vlachos, Dr. Heng-Tze Cheng, Ms. Lavanya Subramanian, from ECE;

Mr. Michael Papamichael, Dr. Frank Lin, Dr. Richard Wang, Mr. Yu-Ting Chen, Mr. Wei-Heng Lo, Mr. Tsung-Hsien Lee, Mr. Vivek Seshadri, Mr. Zhen Tang, Dr. Chulei Liu, Mr. Ilari Shafer, Mr. Mingyu Tang, Mr. Wolfgang Richter, Mr. Mehdi Samadi, Mr. Ankit Sharma from CSD, MLD and other departments.

I would also like to thank Mr. Ross Grieshaber, Ms. Kerry Shihping Chang, Ms. Ruogu Kang, Ms. Cagla Cakir, Ms. Melis Hazar, Mr. Ekin Sumbul, Mr. Berkin Akin, Mr. Burak Erbagci, Mr. Bryan Reiff, for enriching my life both in CMU and in Pittsburgh.

I would like to thank the department of Electrical and Computer Engineering in Carnegie Mellon University, National Science Foundation (NSF), Intel and Samsung, for their generous supports.

Finally, I would like to thank my family−my father, my mother, and my lovely sister. Even though we are separated by the Pacific ocean, I feel you are always standing next to me; your love and support provide me warmth and courage to face every challenge in my life.

To my family

Contents

List of Tables

List of Figures

Chapter 1

Introduction

For the last few decades, the advances in the technology of IC design and manufacturing have led to an explosion in the amount of transistors being integrated. Modeling the performance of computing systems with this ever-growing complexity in a timely manner has become a challenging and critical issue. In this thesis, we propose a learning-based framework to accurately and efficiently model every aspect−performance, power, and operating temperature−of a computing system. The learned models usually possess desired properties, such as convexity and conjugacy[1], for further multi-constrained optimization.

1.1 Performance and energy efficiency

As Moore's Law continues to shrink the feature size of transistors, ever more transistors can now be integrated into a single chip. For the last four decades, this technology scaling has been the fundamental driving force in improving the performance of a computing system. However, in the technology node of 22 nanometer and below, the supply voltage (V_{dd}) and threshold voltage (V_{th}) of a transistor cannot be scaled down proportionally with its feature size [1], resulting in the increase of power density in each process generation of technology scaling. In addition to power density, peak power demands, which are

[1]Conjugacy here refers to the prior and posterior distributions of the same mathematical form. More details can be found in Chapter 2

proportional to the simultaneous switching activities of transistors, also exceed the cooling capability (the ability to dissipate heat) available on current computing systems. Due to these trends, energy efficiency has become a limiting factor in computing performance for all types of platforms, from hand-held devices to mainframe computers deployed in datacenters.

Power density directly translates into heat, and consequently the operating temperature of a processor is getting higher. Excessively high operating temperature is the root of many reliability issues, and can cause temporary timing errors as well as permanent physical damages [2]. The rates of several failure mechanisms, such as aging effects and electromigration, increase exponentially when the operating temperature increases. Furthermore, high operating temperature is known for increasing a processor's power consumption, especially leakage power [2, 3]. The increase of leakage power contributes to the increase of total power consumption, which in turn increases the operating temperature. This thermal-leakage positive feedback loop may lead to thermal runaway phenomenon, and in the worst case may burn the chip.

To scale out from tens of processors (multi-core) to tens of thousands of processors, *i.e.*, the scale of a datacenter, energy efficiency has become an even more severe issue. In addition to the reliability problems mentioned above, energy efficiency directly affects the operation cost (*e.g.*, the cost for operating the cooling system) of a datacenter by approximately 40% [4]. Other performance metric, such as the response latency of a request, also needs to be modeled and considered in order to provide the required Quality of Service (QoS).

1.2 Challenges

To address the issues of energy efficiency, reliability and performance for a multi-core system, a holistic model that can accurately describe the joint impacts of performance, power consumption and thermal hotspots is urgently required. Conventionally, system de-

signers or computer architects rely on full-system (or circuit) simulations [5, 6, 7, 8], to estimate the performance metric of interest[2] for a certain processor design. Although these simulation-based methods usually guarantee a good accuracy in performance modeling, they can be very expensive in terms of execution time (hours to days, per simulation). Since several different models are operated simultaneously inside a simulator and users only have direct control over the inputs, these simulators are often used as "black boxes" by users. In the context of optimization, *e.g.*, architectural design space exploration, designers need to perform exhaustive search over all possible configurations to find the optimal solution, which is almost intractable because of the long simulation time.

The presence of manufacturing process variations, also known as process variations, makes the accurate performance modeling even more challenging. Process variations introduce uncertainties into the performance of each identically-designed processor, such as a magnitude increase in leakage power dissipation [9] and significant performance loss. Therefore, system designers can no longer be isolated from the inherent variability in processor's physical behaviors, such power dissipation and thermal hotspot, and only pursue processor performance. Instead, they need to perform joint optimization for both performance and other crucial physical parameters, for which an analytical model that can enable multi-constrained optimization is urgently required.

1.3 Thesis contributions

In this thesis, we propose **Gray-Box computing**[3], the methodology of a learning-based framework that provides a systematic approach to "learn" an analytical function for quantitatively modeling the performance of a multi-core system. As opposed to conventional simulation-based or physics-based models, Gray-Box computing is a data-driven approach

[2] "Performance of interest" refers to any metric that one is interested in, such as throughput, power consumption or operating temperature.

[3] Gray-Box computing is an extension of the "gray-box model" that is widely-used in machine learning and data mining community. More details are provided in Chapter 2.

that constructs approximation functions, based on the domain knowledge[4] and the underlying patterns discovered from a given dataset. The key idea is to construct the skeleton by using domain knowledge, and specify mathematical details by exploiting the patterns (or regularities) discovered from the data.

1.3.1 Gray-Box overview

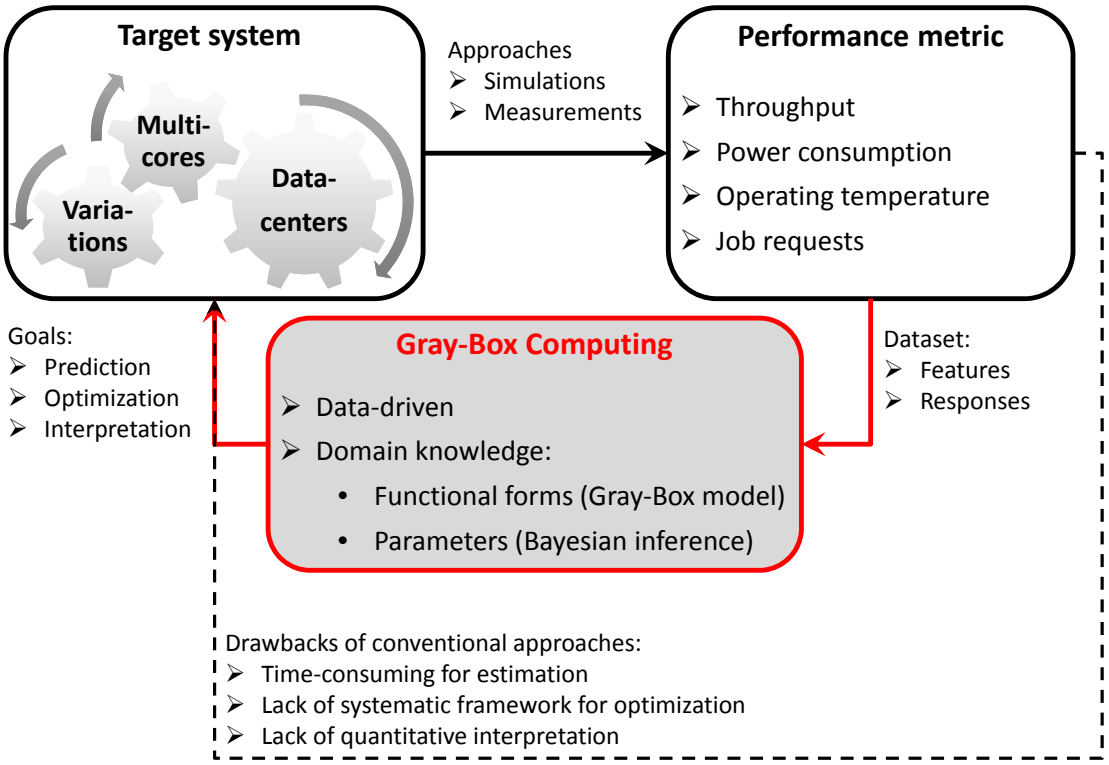


Figure 1.1. Overview of Gray-Box Computing. Gray-Box computing is a data-driven approach that constructs approximation functions based on the domain knowledge and the underlying patterns discovered from a given dataset. Unlike conventional physics-based or simulation-based models that are fixed and used as black-boxes, the flexibility of Gray-Box computing is retained — we can customize the learning procedure from which the functions are trained aiming at a specified goal, such as accurate prediction, multi-constrained optimization, or quantitative interpretation.

[4]In this thesis, the terms *prior domain knowledge*, *prior knowledge* and *domain knowledge* are used interchangeably.

Figure 1.1 presents the overview of Gray-Box computing. Gray-Box computing consists of two components, gray-box model and Bayesian inference, both of which serve as mathematical tools to systematically include one's prior knowledge into model (or function) construction. Since Gray-Box computing is a data-driven approach, it morphs dynamically according to the patterns discovered from the data, resulting in the learned model that can best represent the combination of domain knowledge and data regularities.

Furthermore, the functions learned from Gray-Box computing can be applied to enable a wide range of applications − from accurate prediction, multi-constrained optimization, to quantitative interpretation. Unlike conventional physics-based or simulation-based models that are fixed and used as black-boxes, the flexibility of Gray-Box computing is retained − we can customize the learning procedure from which the functions are trained aiming at a specified goal.

1.3.2 Detailed contributions

In this thesis, we have demonstrated that the proposed Gray-Box computing can be adapted to model and optimize the performance, power consumption and the thermal dynamics of a multi-core systems from a data-driven perspective:

- **Finding the optimal operating point.** We learn a constrained-posynomial model to describe the frequency-power relationship. The accuracy of the learned model is approximately 96%; by leveraging the convexity provided by the learned model, (1) an additional 13.28% of the energy consumption is reduced under iso-performance conditions, and (2) the throughput is increased by additional 7.54% under iso-power conditions.

- **Learning core/uncore cooperative control.** We demonstrate the performance of multi-core systems equipped with on-chip communication fabric can be improved by 10.9% by using reinforcement learning with the parameters learned via Bayesian inference [10].

- **Auto-regression on thermal dynamics.** We show that the learned thermal model, an linear regression with L1 norm regularizer, achieves 98% of accuracy and 113X of speedup compared to conventional architectural simulators [11].

- **Inter-arrival analysis for a datacenter.** We model the job inter-arrivals for a datacenter. The proposed model is able (1) to mimic and interpret multiple components, and (2) to simulate job requests with the same statistical properties as in the real data collected from a large-scale, industrial datacenter.

1.4 Thesis organization

The remainder of this thesis is organized as follows. Chapter 2 details the mathematical formulation of Gray-Box computing. Chapter 3 presents a framework leveraging convexity to find the best operating point for the maximum performance of a multi-core system. Chapter 4 provides a case study of joint optimization of power and performance, and Chapter 5 provides another case study of thermal dynamics modeling, both targeting at also multi-core systems. Scaling out from tens of processors to tens of thousands of processors, Chapter 6 adapts Gray-Box computing for modeling the request inter-arrivals for a datacenter. Chapter 7 provides the related work. Finally, Chapter 8 concludes this thesis and points to possible directions for future research.

Chapter 2

Gray-Box computing

Gray-Box computing is a data-driven, learning-based framework that incorporates prior domain knowledge; the goal is to learn, based on the data, a function $\hat{f}$ that can be used as an approximation of the true function f, when f is unknown, overly-complicated (no closed-form), or difficult to estimate under uncertainties[1]. Furthermore, compared to purely-data-driven approaches such as artificial neural network, Gray-Box computing incorporates prior domain knowledge to learn a function $\hat{f}$ that can be customized to enable certain application(s) of interest, *e.g.*, performing constrained optimization or making inferences by using $\hat{f}$. The domain knowledge is fused into Gray-Box computing via two components: (1) gray-box models and (2) Bayesian inference. In general, gray-box models estimate the functional form of f, and Bayesian inference learns distributions to model underlying uncertainties. These two components are described in Section 2.1 and Section 2.2, respectively.

2.1 Gray-box models

A gray-box model refers to the situation in which, based on the domain knowledge, we know the functional form of f (*e.g.*, a linear function) but the corresponding coefficients are

[1] As a convention in statistics and machine learning community, a symbol with a $\hat{hat}$ represents an estimate instead of the true value or true function.

unknown. Therefore, we need to estimate these coefficients by optimizing a certain objective function. The name gray-box is opposed to *white-box* such as physics-based models or finite-element methods, and opposed to *black-box* such as simulation-based methods or purely-data-driven methods (*e.g.*, artificial neural network). Gray-Box computing is an extension of gray-box models, since compared to gray-box models, we further include Bayesian inference (described in Section 2.2) as another approach to incorporate prior knowledge to facilitate the parameter learning. In the following section (Section 2.1.1), we describe the procedure of estimating function coefficients, and in Section 2.1.2 we elaborate on cross validation (CV), an unbiased error estimator, for calculating errors introduced by the learned model.

2.1.1 Model selection and function estimation

Model selection is a general technique that estimates a function or a set of parameters that best explain the relationship between input features and output responses. Let us assume that a dataset $\mathcal{D}$ is given, where $\mathcal{D}$ contains n samples and each sample consists of a response y and an input vector[2] $\boldsymbol{x}$:

$$\mathcal{D} = \{(y, \boldsymbol{x})_\ell\}, \;\; \ell = 1 \cdots n. \tag{2.1}$$

where $y \in \mathbb{R}$, $\boldsymbol{x} = (x_1, \cdots, x_n)^T \in \mathbb{R}^d$, and ℓ represents the ℓ^{th} sample. Here, y and $\boldsymbol{x}$ are generic representation for the responses and inputs of one's interest. For example, in Chapter 5, y stands for the operating temperature and $\boldsymbol{x}$ is the power consumption (and other features) of a chip-multiprocessor (CMP). Let f be the true function that maps $\boldsymbol{x}$ to y:

$$y = f(\boldsymbol{x}) + \epsilon \tag{2.2}$$

[2]Bold fonts represent vectors or matrices instead of scalars. Also in this thesis, vectors refer to column vectors.

ϵ is a random variable (usually treated as a random error or noise) that follows a certain distribution $\mathcal{P}$, denoted as $\epsilon \sim \mathcal{P}$. In many cases, true f is either unknown, difficult to derive (no analytical form), or not suitable for further optimization due to its complexity. Therefore, we adapt the model selection technique to learn $\hat{f}$, as the estimate of f, directly from the dataset of interest. The procedure of learning $\hat{f}$ is called *training phase*, as opposed to *validation (or testing) phase* in which the accuracy of the learned $\hat{f}$ is examined.

In the context of gray-box models, we know the approximate (or exact) functional form of f, based on the domain knowledge, and the coefficients remain unknown. Without losing generality, here we use a linear function as an example:

$$f(\boldsymbol{x}; \boldsymbol{\alpha}) = \boldsymbol{\alpha}^T \boldsymbol{x} = \sum_{i=1}^{d} \alpha_i x_i \tag{2.3}$$

$f(\boldsymbol{x}; \boldsymbol{\alpha})$ denotes that $\boldsymbol{\alpha}$ are parameters of f. Now, the problem of learning f is converted into searching for the best estimate of $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)^T$, denoted as $\hat{\boldsymbol{\alpha}}$. In general, $\hat{\boldsymbol{\alpha}}$ can be obtained by minimizing a loss function, $L(\boldsymbol{\alpha}; \boldsymbol{x})$:

$$\hat{\boldsymbol{\alpha}} = arg\min_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}; \boldsymbol{x}) \tag{2.4}$$

Note that $L(\boldsymbol{\alpha}; \boldsymbol{x})$ is a function of $\boldsymbol{\alpha}$ (the parameters used in f), not $\boldsymbol{x}$ (samples). One widely-used loss function is *squared error*, which can be plugged into Eq(2.4):

$$\begin{aligned} \hat{\boldsymbol{\alpha}} &= arg\min_{\boldsymbol{\alpha}} \sum_{\ell=1}^{n} \Big(y_\ell - f(\boldsymbol{x}_\ell; \boldsymbol{\alpha}) \Big)^2 \\ &= arg\min_{\boldsymbol{\alpha}} \sum_{\ell=1}^{n} \Big(y_\ell - \boldsymbol{\alpha}^T \boldsymbol{x}_\ell \Big)^2 \end{aligned} \tag{2.5}$$

The objective function in Eq(2.5) is a quadratic function, and therefore it is a convex program where the optimal values of $\hat{\boldsymbol{\alpha}}$ can be found efficiently [12]. Here, no constraint is

specified in Eq(2.5). After $\hat{\boldsymbol{\alpha}}$ is calculated, $\hat{f}$ can be constructed by plugging $\hat{\boldsymbol{\alpha}}$ back into the original f. This is the well-known linear regression with least-square-fit [13].

By examining the procedure of learning a linear function, we raise an intriguing question: can this procedure be customized to learn a function $\hat{f}$ that possesses certain desired properties? One example of such properties can be sparsity, *i.e.*, most entities in $\hat{\boldsymbol{\alpha}}$ are zero, and another property can be the convexity embedded in a non-linear $\hat{f}$. As we will show in Chapter 3 and 5, $\hat{f}$ with desired properties can be learned by customizing the functional form, loss function, or constraints specified to optimization programs such as Eq(2.5).

2.1.2 Cross validation

In the validation phase, the accuracy of the learned model $\hat{f}$ is examined. Cross validation (CV) is an unbiased error estimator and widely-used in machine learning and statistics to avoid over-fitting [14]. Over-fitting refers to the situation that the learned model $\hat{f}$ fits the training samples by using an overly-complex model, and has poor predictive accuracy on the "clean" samples, *i.e.*, the data not used to learn $\hat{f}$. The purpose of the validation phase is twofold. First, we want to examine if $\hat{f}$ is consistent[3] with f. In other words, $\hat{f}$ should have "very similar behaviors" as f, both qualitatively and quantitatively, over the domain of f. Second, CV facilitates the process of model selection and avoids the pitfall of over-fitting. In Section 2.1, we select functional forms for learning $\hat{f}$ based on prior knowledge. However, this may not be sufficient in certain cases. For example, we want to include new features into $\hat{f}$, *i.e.*, increasing the dimensionality of $\boldsymbol{x} \in \mathbb{R}^d$ to $\mathbb{R}^{d'}$ where $d \leq d'$. In this context, the error reported by CV can be used as an indicator to determine the best $d' -$ the d' that leads to the smallest error reported by CV is selected.

Here, we introduce leave-one-out cross validation (LOOCV) to estimate the error of the learned model $\hat{f}$. LOOCV involves using a single sample from the original dataset as the validation (clean) data, and the remaining samples as the training data used to learn $\hat{f}$

[3]According to [13, 14], the consistency is defined as $\mathbb{P}(|f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})| \geq \delta) \to 0$ when the sample size $n \to \infty$. δ is a sufficiently-small positive number.

during the training phase. The process is repeated such that each sample in the dataset is used exactly once as the validation data. The root-mean-squared-error (RMSE) reported by LOOCV can be expressed as:

$$RMSE = \left[\frac{1}{n}\sum_{\ell=1}^{n}\left(y_\ell - \hat{f}_{\setminus\ell}(\boldsymbol{x}_\ell)\right)^2\right]^{\frac{1}{2}} \tag{2.6}$$

where $\hat{f}_{\setminus\ell}$ represents the model learned without the ℓ^{th} sample during the training phase. The procedure of LOOCV can also be applied for calculating other error metrics, such as mean-square-error (MSE) or root-mean-square-percentage-error (RMSPE) in Chapter 3. An alternative to perform LOOCV is "k-fold cross validation", in which the original sample is randomly partitioned into k equal-size subsets. A single subset, out of the k subsets, is retained as the validation data for testing the model, and the remaining $(k-1)$ subsets are used as training data. The procedure of k-fold cross validation is highly similar with LOOCV and hence not described here.

2.2 Bayesian inference

Bayesian inference, also known as Bayesian statistics or Bayesian learning, is a statistical approach that incorporates one's domain knowledge to estimate a probabilistic distribution that describe uncertainties. Bayesian inference views each parameter as a random variable, and applies Bayes' rule to update the probability (or called belief) of a learned parameter as additional evidences are observed.

2.2.1 Bayes rule and prior distribution

Formally, Bayesian inference computes the posterior probability as a consequence of a *prior probability* and a *"likelihood function"* according to Bayes' rule:

$$\begin{aligned} \mathbb{P}(\boldsymbol{\theta}|\boldsymbol{D}) &= \frac{\mathbb{P}(\boldsymbol{D}|\boldsymbol{\theta}) \times \mathbb{P}(\boldsymbol{\theta})}{\mathbb{P}(\boldsymbol{D})} \\ &\propto \mathbb{P}(\boldsymbol{D}|\boldsymbol{\theta}) \times \mathbb{P}(\boldsymbol{\theta}) \end{aligned} \tag{2.7}$$

where $\boldsymbol{D}$ is the data observed and $\boldsymbol{\theta}$ is a set of parameters that describe the probability distribution of observed data. $\mathbb{P}(\boldsymbol{\theta}|\boldsymbol{D})$ is known as the *posterior probability*, $\mathbb{P}(\boldsymbol{D}|\boldsymbol{\theta})$ represents the data likelihood and $\mathbb{P}(\boldsymbol{\theta})$ is the prior probability. The normalization term $1/\mathbb{P}(\boldsymbol{D})$ is a constant to force $\mathbb{P}(\boldsymbol{\theta}|\boldsymbol{D}) \in [0, 1]$ and in general does not affect parameter estimates.

2.2.2 Conjugate prior and posterior

The update of the posterior probability is usually an iterative process as shown in Figure 2.1. First we determine the prior probability $\mathbb{P}(\boldsymbol{\theta})$ based on our domain knowledge, and the posterior probability $\mathbb{P}(\boldsymbol{\theta}|\boldsymbol{D})$ can be calculated by Eq(2.7), *i.e.*, the multiplication of data likelihood $\mathbb{P}(\boldsymbol{D}|\boldsymbol{\theta})$ and prior $\mathbb{P}(\boldsymbol{\theta})$. In the second iteration, the updated posterior is fed back as a new prior while the data likelihood is also updated since more data are observed. Therefore, the new posterior can be calculated accordingly.

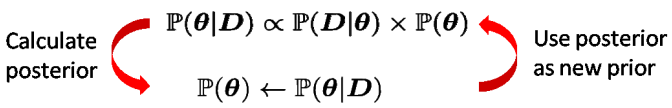


Figure 2.1. The iterative update of prior, data likelihood and posterior.

After the posterior $\mathbb{P}(\boldsymbol{\theta}|\boldsymbol{D})$ is obtained, we select $\hat{\boldsymbol{\theta}}$ that maximizes $\mathbb{P}(\boldsymbol{\theta}|\boldsymbol{D})$ as the estimate of true $\boldsymbol{\theta}$. $\hat{\boldsymbol{\theta}}$ is called Maximum A Posteriori (MAP) estimate. Like we mentioned earlier, the prior distribution is determined based on the domain knowledge. A good selection of prior expedites the convergence of $\hat{\boldsymbol{\theta}} \rightarrow \boldsymbol{\theta}$. Even if the prior is not selected

properly, MAP always guarantees the convergence of $\hat{\boldsymbol{\theta}} \rightarrow \boldsymbol{\theta}$ with sufficient amount of data observed. As a comparison to maximum likelihood estimate (MLE) that selects $\hat{\boldsymbol{\theta}}$ to maximize the data likelihood $\mathbb{P}(\boldsymbol{D}|\boldsymbol{\theta})$, MAP employs an augmented optimization by using a prior distribution over the parameters of interests, $\boldsymbol{\theta}$ in this case. Generally, MAP leads to a more accurate and robust estimate than MLE [14].

Chapter 3

Learning the optimal operating point for multi-core systems

Dynamic voltage and frequency scaling (DVFS) is one of the widely-used methods to increase the energy efficiency of a multi-core system. The selection of voltage and frequency is referred to the "operating point." To maximize the energy efficiency, we manage to find the optimal operating point for (1) the maximum performance under the power constraint and (2) the minimum energy consumption under the performance constraint. In this chapter, we apply Gray-Box computing along with model selection techniques to learn a model that leverages convexity for multi-constrained optimization.

3.1 Operating points from near-threshold regime to turbo boost

Near-threshold computing (NTC) is a promising design methodology to achieve high energy efficiency in scaled CMOS technology nodes. In the NTC regime, the supply voltage (V_{dd}) of processors is reduced to a value near the threshold voltage (V_{th}) in order to reduce energy consumption at the cost of frequency degradation [15]. It has been shown that NTC can bring an order of improvement in energy efficiency over conventional designs operated at the nominal supply voltage [16]. However, the significant performance degradation due

to this aggressive voltage down-scaling may be undesirable for performance-constrained applications, since the overall performance may drop significantly.

On the other hand, upscaling of voltage/frequency of a processor above nominal settings has been explored recently by processor manufacturers, *i.e.*, Turbo Boost[1] (TB) [17], for improving the performance [18]. This technique is also referred to as Reverse Dynamic Voltage Frequency Scaling (RDVFS) by others [10]. The goal of RDFVS is to maximize performance within a predefined power budget, as opposed to the traditional DVFS goal of maximizing the performance per unit of power consumption.

3.1.1 Maximum energy efficiency

We conjecture that, to maximize energy efficiency while meeting user specified demands or constraints, modern processors and multi-core systems have to be equipped with a wide range of voltage and frequency values that include TB, nominal and NTC modes of operation. A processor prototype with these capabilities has been recently demonstrated [19]. However, given the variations within/across multi-threaded workloads, the benefits of the extended range DVFS for multi-core systems have not been quantified. In this context, an important question to address is to determine the maximum benefit that can be achieved by performing DVFS over a wide-operating range that includes TB, nominal and NTC modes, while satisfying predefined constraints.

To answer this question, system designers and computer architects need a systematic framework that fulfills the following two requirements: (1) the capability to accurately model the power versus performance curve over a wide operating range in an efficient, analytic framework, and (2) efficient optimization techniques that leverage the analytical model to determine the optimal operating voltage and frequency pair for each core under user-specified constraints. However, the performance and power characteristics in TB, nominal and NTC regimes are distinct, so it is difficult to analytically derive a one-size-fits-

[1]Turbo Boost technology has been implemented as a standard operation of Intel® Itanium® platforms.

all model. Furthermore, the optimization techniques also need to be carefully designed to work seamlessly with the model. This interdependency adds extra challenges to the problem of selecting the best voltage-frequency pair that leads to the maximum performance (or energy) benefit under predefined constraints.

3.1.2 Contributions and chapter organization

To the best of our knowledge, the following novel contributions are described and supported in this thesis:

- We propose a constrained-posynomial[2] model to learn the frequency-power relationship, based on which leave-one-out cross validation (LOOCV) [20] is applied to select the best parameters. Experimental results demonstrate that the proposed learning framework achieves a *nearly-consistent* [21] model with a root-mean-squared-percentage-error (RMSPE) of only 4.37%.

- By leveraging the convexity of the learned model, we convert the problem of the energy minimization under performance requirements into a constrained convex optimization problem that can be solved efficiently. The optimal operating frequency can then be obtained via well-established algorithms for solving convex optimization problems. We further perform a sensitivity test over various performance requirements to examine the best energy efficiency that can be achieved by NTC and TB.

- As opposed to energy minimization, we also determine the maximum possible performance under a given power budget, *i.e.*, the optimality of RDVFS over a wide-operating range. This optimization is also enabled by the convexity of the learned model.

- We evaluate the proposed learning-based optimization framework with a wide spectrum of parallel, multi-threaded applications [22][23] as well as synthetic bench-

[2]The mathematical definition of constrained-posynomial is provided in Eq(3.10), Section 3.3.2.

marks executed on a full-system simulator [24]. The experimental results confirm the effectiveness of simultaneously using NTC and TB: (1) on average **additional** 13.28% of the energy consumption is reduced under iso-performance conditions, and (2) on average throughput is increased by **additional** 7.54% under iso-power conditions, both compared with DVFS in the conventional operating range.

The organization of this chapter is as follows. Section 3.2 introduces the necessary background on near-threshold operations. Section 3.3 details the proposed model learning process. Section 3.4 presents the constrained optimization framework by using the learned power-performance model. Section 3.5 provides the overall implementation flow. Section 3.6 demonstrates the experimental results, while Section 3.7 provides the discussion.

3.2 Near-threshold operation & performance metric

In this section, we provide (1) the necessary background of NTC, including both delay and power calculations, and (2) the performance metric used in this chapter.

3.2.1 Delay model

When transistors are operated in the near-threshold region, the Enz-Krummenacher-Vittoz model [25] is shown to be very accurate for modeling the electrical current [26]. At the onset of inversion, the current flowing through the drain and source of a transistor I_{ds} can be expressed as :

$$I_{ds} = \left(2 \cdot n \cdot \mu \cdot C_{ox} \cdot \frac{W}{L} \cdot \phi_t^2\right) \cdot \frac{IC}{k_{fit}} \tag{3.1}$$

where n is the sub-threshold slope, μ is the mobility, C_{ox} is the oxide thickness, W and L are the gate width and channel length, ϕ_t^2 is the thermal voltage, IC is the inversion coefficient and k_{fit} is a fitting coefficient [26]. Let V_{gs} represent the potential difference between the gate and source of a transistor; typically $IC > 10$ when $V_{gs} \gg V_{th}$, $0.1 <$

$IC < 10$ when V_{gs} is close to V_{th} (near-threshold), and $IC \leq 0.1$ when $V_{gs} < V_{th}$ (sub-threshold).

With I_{ds}, the critical path delay (D_{crit}) can be expressed as:

$$D_{crit} = k_{tech} \cdot k_{str} \cdot N_L \cdot \frac{(k_g \cdot C_L \cdot V_{dd})}{I_{ds}} \tag{3.2}$$

where N_L represents the number of logic levels of the critical path, and k_{tech}, k_{str} are technology- and circuit structure-dependent constants. k_g is the fitting coefficient for gate delay and C_L is the output load. The maximum operating frequency can be calculated as $1/D_{crit}$.

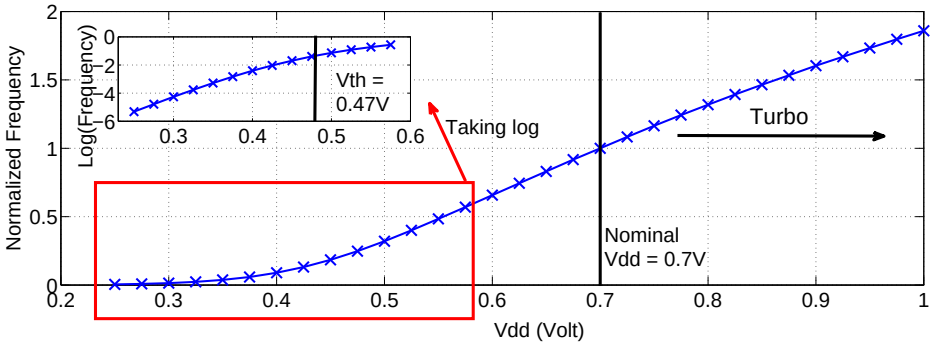


Figure 3.1. Operating frequency under a wide range of V_{dd}

Fig.3.1 shows the maximum operating frequencies under a wide range of V_{dd}. The frequency is measured by using a seven-stage fan-out-of-four (FO4) ring oscillator (RO) with 16nm predictive technology model (PTM) [27]. The frequency of an RO has been used as an accurate proxy for the operating frequency of a processing core in prior work [28]. From Fig.3.1, we can observe that the curve is approximately linear around nominal and turbo regions, but sub-exponential in the near-threshold region. A similar trend was also reported by [19]. Due to the distinct characteristics of these three regions, it is difficult to derive a one-size-fits-all analytical model that can accurately describe the operating frequencies over a wide range of V_{dd}.

3.2.2 Energy model

The total energy (E_g) consumption per gate in every clock period can be separated into dynamic energy (E_{dyn}) and leakage energy (E_{leak}):

$$E_g = E_{dyn} + E_{leak} = C_L \cdot V_{dd}^2 \cdot \alpha_s + I_{leak} \cdot V_{dd} \cdot D_{crit} \tag{3.3}$$

where α_s is the average per gate switching activity, *i.e.*, D_{crit} as expressed by Eq(3.2). I_{leak} is the leakage current per gate, which equals $\frac{I_{ds} \cdot k_{fit}}{IC}$. All I_{ds}, k_{fit} and IC are described in Eq(3.1). Total energy of a logic module can be expressed as:

$$E_{tot} = k_{str}^{dyn} \cdot \left(C_L \cdot V_{dd}^2 \cdot \alpha_s\right) + k_{str}^{leak} \cdot \left(I_{leak} \cdot V_{dd} \cdot D_{crit}\right) \tag{3.4}$$

where k_{str}^{dyn} is proportional to the number of switching gates of a module, and k_{str}^{leak} is a function of the total number of devices per module, the fraction of OFF devices and the stacking effect.

3.2.3 Performance metric

A key metric to measure the performance of a chip-multiprocessor (CMP) is the total throughput ($\mathcal{T}$), *i.e.*, the number of instructions executed per unit of time over all cores in the CMP. Given the instructions executed per cycle ($\mathcal{I}$) of each processing core, the throughput can be calculated as:

$$\mathcal{T} = \sum_{i=1}^{n} \mathcal{I}_i \cdot \mathfrak{F}_i \tag{3.5}$$

where $\mathfrak{F}_i$ and $\mathcal{I}_i$ are the operating frequency and IPC of core i, respectively. n is the number of cores in the CMP. According to [29], the instructions executed per cycle ($\mathcal{I}$) can be expressed as:

$$\mathcal{I} = \left[\mathcal{C}_{comp} + \rho \cdot \tau\right]^{-1} \tag{3.6}$$

where $\mathcal{C}_{comp}$ is the number of clock cycles required for executing an instruction with an ideal last-level cache (LLC), ρ is the average number of LLC misses per instruction, and τ is the cache miss penalty, which can be calculated by the average cycles spent per LLC miss. Eq(3.6) can further be extended to model other performance overheads, such as L1 cache misses and on-chip communication latency, by including additional parameters. In this chapter, we focus on the effects of LLC and the extension of Eq(3.6) is left as a future work. Note that τ is measured by using the uncore[3] frequency ($\mathfrak{F}_{uc}$). When the frequency of a core is scaled and differs from the uncore frequency ($\mathfrak{F}_{uc}$), τ needs to be correspondingly adjusted by multiplying the ratio of the core and the uncore frequency. Therefore, Eq(3.6) can be further modified as:

$$\mathcal{I} = \big[\mathcal{C}_{comp} + (\rho \cdot \tau) \cdot \frac{\mathfrak{F}}{\mathfrak{F}_{uc}}\big]^{-1} \tag{3.7}$$

In this chapter, we assume the frequency of each core can be scaled, whereas the frequency of uncore is fixed to its nominal value. As Fig.3.1 shows, the frequency is defined as the maximum possible frequency under a given supply voltage. For conciseness, in the remainder of this chapter, when we mention the change of frequency, we actually refer to the change of both frequency and voltage, *i.e.*, the frequency-voltage pairs.

3.3 Model learning process

The proposed power model has two components: (1) the frequency function f, and (2) the utilization function u. To learn $\hat{f}$ and $\hat{u}$, we adapt constrained-posynomial functions to map both operating frequency and utilization to the corresponding power consumption by using leave-one-out cross validation (LOOCV). The accuracy and overall validation of the learned model are also provided.

[3]In this chapter we refer to uncore as representing the last-level cache and on-chip communication fabric.

3.3.1 Proposed power model

To study the best possible benefit of applying DVFS over a wide-operating range under workload variations, we first learn a model that accurately characterizes the power consumption as a function of the operating frequency under different workload behaviors. Changing the voltage and frequency of a core impacts its power consumption in two ways. First, as observed in Eq(3.3) and Eq(3.4), the dynamic and static power consumption of each transistor depend on the operating voltage and frequency. Second, the instructions executed per second depend on the operating frequency (as seen in Eq(3.7)) and therefore the utilization of a core changes with the frequency. Thus, the power consumption of a processing core as a function of operating frequency, $\mathcal{P}(\mathfrak{F})$, can be written as:

$$\mathcal{P}(\mathfrak{F}) = P_{dyn}^{peak} \cdot f_d(\mathfrak{F}) \cdot u_{\mathcal{I}}(\mathfrak{F}) + P_{sta}^{peak} \cdot f_s(\mathfrak{F}) \tag{3.8}$$

The equation above consists of three components: (1) P_{dyn}^{peak} and P_{sta}^{peak} are the peak dynamic and static power consumption of each core from the design specifications; (2) $f_d(\mathfrak{F})$ and $f_s(\mathfrak{F})$ encapsulate the impact of voltage and frequency scaling on the peak dynamic and static power consumption, assuming the the processor utilization remains constant; (3) $u_{\mathcal{I}}(\mathfrak{F})$ encapsulates the change in processor utilization as a result of frequency scaling. Implicitly, $\mathcal{I}$ is a function of $\mathfrak{F}$ as Eq(3.7) shows, and therefore we denote $u_{\mathcal{I}}(\mathfrak{F})$ with a subscript of $\mathcal{I}$ to express that the utilization is a function of $\mathfrak{F}$ through $\mathcal{I}$.

From Eq(3.8), except P_{dyn}^{peak} and P_{sta}^{peak} which can be obtained from the design specification, all three other components need to be learned from empirical data: $f_d(\mathfrak{F})$, $f_s(\mathfrak{F})$ and $u_{\mathcal{I}}(\mathfrak{F})$. In general, $f_d(\mathfrak{F})$ and $f_s(\mathfrak{F})$ model the respective changes of dynamic and static power consumption due to voltage/frequency scaling from turbo to near-threshold regimes (workload-independent), whereas $u_{\mathcal{I}}(\mathfrak{F})$ models the dynamic power changes due to workload characteristics.

3.3.2 Learning frequency-power relationship

In this section, we focus on learning $f_d(\mathfrak{F})$ and $f_s(\mathfrak{F})$. Since we adapt the same methodology to learn both $f_d(\mathfrak{F})$ and $f_s(\mathfrak{F})$, we focus on only $f_d(\mathfrak{F})$ here and denote it as f for conciseness. Let $p = f(\mathfrak{F})$, where p is the power consumption and $\mathfrak{F}$ is the operating frequency. The physical interpretation of f is that p represents the amount of power consumed when a processor (or a processing core) is operating at the frequency of $\mathfrak{F}$ (assuming constant utilization). All values of p are normalized to the power consumed at the nominal frequency, $\mathfrak{F}_{nom}$. f can be a complicated function that depends on the technology node, design details, and other factors including the terms in Eq(3.1)$-$(3.4).

One approach for determining the relationship between frequency and power consumption is to use physics-based models, such as those used in circuit simulators (*e.g.*, HSPICE®). However, to achieve high accuracy, different models must be used in different modes of operation. This makes it difficult to determine a *unified*, *convex* analytical function that spans the wide range of frequency/voltage values ranging from the TB to NTC modes of operation, which is required to efficiently determine the optimal operating point and is the goal of this chapter. Therefore, as an alternative to a physics-based model, we propose to *learn* a less complex function $\hat{f}$ that behaves like f.

The model-learning procedure can be separated into two phases: *training phase* and *validation phase*. The goal of the training phase is to learn the model $\hat{f}$, while in the validation phase the accuracy of the learned model is examined. Both phases require a dataset with samples that contain the power consumption under a certain frequency, denoted as $(p^\ell, \mathfrak{F}^\ell)$ for $\ell = 1, \ldots, s$, where s is the total number of samples. This dataset can be obtained from simulations or measurements, such as using simulations or measurements from chip prototypes. As we will show later in Section 3.3.2, the proposed learning framework achieves high accuracy in the datasets obtained from simulations and from the industrial measurements provided by [19].

Training phase

In general, the power consumption p is a monotonically-increasing function of $\mathfrak{F}$ – operating a processor at a higher frequency consumes more power when the other conditions are fixed. We approximate this monotonically-increasing function by using a *constrained-posynomial* function. By selecting a proper posynomial degree, a posynomial is *convex*, based on which no local minimum exists, or any local minimum equals the global minimum.

The algebraic expression of a general posynomial can be expressed as:

$$f(\mathfrak{F}) = \sum_{i=0}^{d} \alpha_i \mathfrak{F}^{\beta_i} = \alpha_0 \mathfrak{F}^{\beta_0} + \alpha_1 \mathfrak{F}^{\beta_1} + \cdots + \alpha_d \mathfrak{F}^{\beta_d} \tag{3.9}$$

where $\alpha_i \geq 0$, $\beta_i \in \mathbb{R}$, $d \in \mathbb{N}$, $\mathfrak{F} > 0$, and $\mathfrak{F}^{\beta_i}$ represents the β_i^{th} power of $\mathfrak{F}$. Unlike general posynomials that allow $\beta_i \in \mathbb{R}$, here we enforce $\beta_i \in \mathbb{Z}_+$ (non-negative integers) and define such posynomials as *constrained-posynomials*:

$$f(\mathfrak{F}) = \sum_{i=0}^{d} \alpha_i \mathfrak{F}^{i} = \alpha_0 \mathfrak{F}^{0} + \alpha_1 \mathfrak{F}^{1} + \cdots + \alpha_d \mathfrak{F}^{d} \tag{3.10}$$

where all symbols are defined as in Eq(3.9). Since $f''(\mathfrak{F}) \geq 0$, constrained-posynomials are always convex (and strongly convex in our case). Now the problem of learning f is converted into searching for the optimal $\boldsymbol{\alpha} = (\alpha_0, \ldots \alpha_d)^T$ and d. Bold fonts here represent vectors or matrices instead of scalars. Under any given $d \in \mathbb{N}$, $\hat{\boldsymbol{\alpha}}$ can be obtain via solving the following equations:

$$\hat{\boldsymbol{\alpha}} = arg\min_{\boldsymbol{\alpha}} \left\{ \sum_{\ell=1}^{s} \left(p^{\ell} - \sum_{i=0}^{d} \alpha_i (\mathfrak{F}^{\ell})^{i} \right)^2 \right\} \tag{3.11}$$

$$s.t.\ \alpha_i \geq 0 \text{ , for } i = 0 \ldots d \tag{3.12}$$

where p^{ℓ} and $\mathfrak{F}^{\ell}$ are the power consumption and frequency of the ℓ^{th} sample, and s is the total number of samples. Eq(3.11) is a squared loss function which is commonly used as the objective to be minimized for curve fitting [30][31][11]. For each α_i, the objective function in Eq(3.11) is quadratic, and all d constraints in Eq(3.12) are linear. Therefore, Eq(3.11)(3.12) together forms a convex program, whose optimal solution $\hat{\boldsymbol{\alpha}}$ can be calculated efficiently by using gradient methods or other algorithms [31].

Validation phase

In the validation phase, the accuracy of the learned model $\hat{f}$ is examined, which in turn helps in determining the best d in Eq(3.10). We apply LOOCV to estimate the error of the learned model $\hat{f}$. LOOCV is an unbiased error estimator and widely-used in machine learning and statistics to avoid over-fitting [20]. Over-fitting refers to the situation in which the learned model $\hat{f}$ fits the training samples by using an overly-complex model, such as a high-order constrained-posynomial, and has poor predictive accuracy on the "clean" data, *i.e.*, the data not used to learn $\hat{f}$. Therefore, the best d in Eq(3.10) can be obtained by selecting the value that leads to the smallest error reported by LOOCV. LOOCV involves

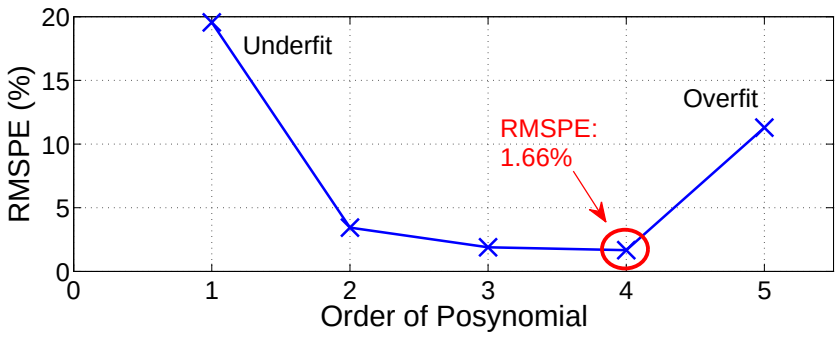


Figure 3.2. Cross validation error *vs.* order of constrained posynomial

using a single sample from the original dataset as the validation (clean) data, and the remaining samples as the training data used to learn $\hat{f}$ during the training phase. The process is repeated such that each sample in the dataset is used exactly once as the validation data.

The root-mean-squared-percentage-error (RMSPE) reported by LOOCV can be expressed as:

$$RMSPE = \left[\frac{1}{s}\sum_{\ell=1}^{s}\left(\frac{p^{\ell} - \hat{f}_{\setminus\ell}(\mathfrak{F}^{\ell})}{p^{\ell}}\right)^{2}\right]^{\frac{1}{2}} \tag{3.13}$$

where $\hat{f}_{\setminus\ell}$ represents the model learned without the ℓ^{th} sample during the training phase. Fig.3.2 shows the RMSPE of $\hat{f}$ with $d = 1 \ldots 5$. The $\hat{f}$ with $d = 4$ achieves the smallest RMSPE of 1.66%, and other values of d yield larger RMSPEs, since either underfitting or overfitting may occur when an overly-small or overly-large d is selected. The trend persists when $d > 5$ (the maximum d considered in this chapter is 10). With LOOCV and Eq(3.11)(3.12), we have demonstrated a systematic framework to learn the best $\hat{f}$ for a given dataset.

Dataset for learning $\hat{f}_d(\mathfrak{F})$ and $\hat{f}_s(\mathfrak{F})$

To learn $\hat{f}_d$, a dataset of dynamic power consumption with a fixed utilization at different frequencies, denoted as $(p^{\ell}, \mathfrak{F}^{\ell})$, is required as the input for the model-learning procedure (with LOOCV) described in Section 3.3.2 and 3.3.2. Since the utilization of an odd-stage ring oscillator is always fixed, we measure $(p^{\ell}, \mathfrak{F}^{\ell})$ by using a seven-stage fan-out-of-four (FO4) ring oscillator with 16nm predictive technology model (PTM) [27]. The V_{dd} is ranging from 0.25V to 1V with the nominal V_{dd} = 0.7V and the V_{th} = 0.47V as depicted in Fig.3.1. According to [19][16], the operation modes are defined as follows: (1) Turbo mode: $V_{dd} = [0.7, 1.0]$ and $\mathfrak{F} = [1, 1.86]$, (2) nominal mode: $V_{dd} = [0.55, 0.7]$ and $\mathfrak{F} = [0.48, 1]$, (3) near-threshold mode: $V_{dd} = [0.37, 0.55]$ and $\mathfrak{F} = [0.059, 0.48]$. Note that all $\mathfrak{F}$ input to our framework are normalized w.r.t. the nominal case. The resulting $(p^{\ell}, \mathfrak{F}^{\ell})$ are shown in Fig.3.3, and all the measurements are normalized to the frequency and power of $V_{dd} = 0.7$ to show the relative trend. Similarly, we collect the static power consumption from an inverter-loop with the same technology setting described above for learning $\hat{f}_s$.

Accuracy of the learned $\hat{f}_d(\mathfrak{F})$ and $\hat{f}_s(\mathfrak{F})$

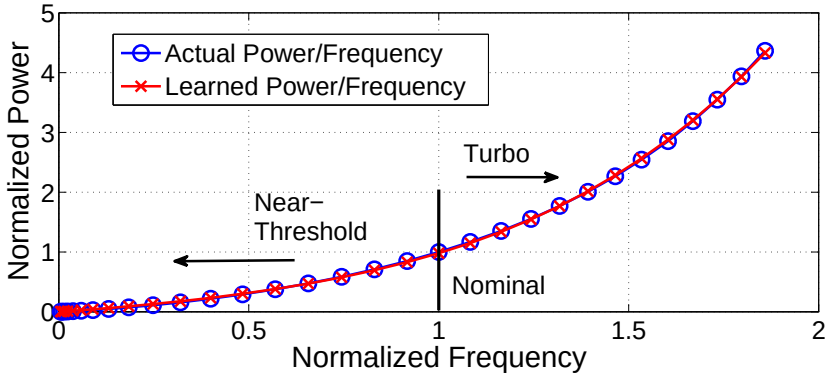


Figure 3.3. Accuracy of learned frequency-power function

We illustrate the accuracy of $\hat{f}$ in Fig.3.3. The actual frequency-power curve obtained from the simulations, f, is delineated in red, whereas the one predicted from $\hat{f}$ is delineated in blue. The implementation details are described in Section 3.5. As Fig.3.3 shows, the learned $\hat{f}$ behaves almost exactly like the actual curve, and achieves a very high accuracy of 98.34% (a RMSPE of 1.66%). Therefore, $\hat{f}$ is described as *nearly-consistent* with f, since its RMSPE is close to zero and its behavior is consistent with the actual model [30]. Finally, we repeat the complete procedure described in Section 3.3.2 and 3.3.2 to learn $f_s(\mathfrak{F})$ and obtain highly simiar results: the RMSPE is 1.21% and the d is 6. To further evaluate the proposed model-learning framework, we learn the $\hat{f}$ from the actual frequency-power measurements provided by [19]. The result is qualitatively and quantitatively similar: the learned $\hat{f}$ achieves a RMSPE of only 1.32%. Both results demonstrate the robustness and generality of the proposed model-learning framework.

3.3.3 Learning utilization-power relationship

After $\hat{f}_d$ and $\hat{f}_s$ are obtained, we now learn the utilization-power relationship, $u_{\mathcal{I}}(\mathfrak{F})$. As prior work (such as [32][33]) has pointed out, the dynamic power can be approximated as a linear function of IPC, since IPC approximately represents the activity rate of a processing

core. Therefore, $u_{\mathcal{I}}(\mathfrak{F})$ can be expressed as:

$$u_{\mathcal{I}}(\mathfrak{F}) = c_1 \cdot \mathcal{I} + c_2 \tag{3.14}$$

where c_1 and c_2 are fitting coefficients. From the dataset, a positive correlation between IPC and dynamic power consumption has been observed. In other words, higher IPC contributes to higher dynamic power dissipation, and vice versa, which leads to a positive slope c_1. Furthermore, c_2 is also positive since the power is still consumed even for a very low (or close to zero) IPC. Therefore, both c_1 and c_2 are positive and the convexity of model is maintained. Eq(3.14) is a constrained posynomial with $d = 1$, and we repeat the learning procedure described in Section 3.3.2 and 3.3.2 to learn $\hat{c}_1$, $\hat{c}_2$ with d set to 1.

Here, we describe the dataset $(p^\ell, \mathcal{I}^\ell)$ used learn the $\hat{c}_1$, $\hat{c}_2$ of $\hat{u}_{\mathcal{I}}(\mathfrak{F})$. As opposed to the dataset described in Section 3.3.2, the operating frequency of $(p^\ell, \mathcal{I}^\ell)$ is fixed at the nominal value, and therefore the changes of dynamic power dissipations are only from the workload characteristics, not voltage/frequency scalings. We use Sniper [24] as the architectural simulator to collect IPCs ($\mathcal{I}^\ell$) and other required workload characteristics ($\mathcal{C}_{comp}$, ρ, τ). The target architecture is described in Table 3.1. Default settings are used for the parameters not mentioned here. For benchmarks, we use both PARSEC [22] and SPLASH-2 [23] that contain a wide spectrum of multi-threaded parallel applications. For calculating power, we use McPAT [34] to collect dynamic power traces (p^ℓ). Finally, the average error of the learned is $\hat{u}_{\mathcal{I}}(\mathfrak{F})$ 5.60%. Similar results are also reported by others [32][33]. After $\hat{f}_d(\mathfrak{F})$, $\hat{f}_s(\mathfrak{F})$ and $\hat{u}_{\mathcal{I}}(\mathfrak{F})$ are obtained, we are in position to examine the overall learned model $\hat{P}(\mathfrak{F})$.

3.3.4 Model validation

Previously, the accuracies of $\hat{f}_d(\mathfrak{F})$, $\hat{f}_s(\mathfrak{F})$ and $\hat{u}_{\mathcal{I}}(\mathfrak{F})$ are validated separately, and therefore the accuracy of the overall learned $\hat{\mathcal{P}}(\mathfrak{F})$ remains unknown. In this section, $\hat{\mathcal{P}}(\mathfrak{F})$ is validated with the power consumption of a whole processor at different frequencies under

Table 3.1. Target architecture

Parameters	Values
Number of cores	16
Nominal frequency	2660 MHz
Core model	Intel®-X86 Gainestown®
L2 caches	Private 256KB, 4-way SA, LRU
L3 caches	Shared 32MB, 16-way SA, LRU
DRAM	4GB
Technology	16nm node with nominal $V_{dd} = 0.7$V

various workload characteristics. By plugging $\hat{f}_d(\mathfrak{F})$, $\hat{f}_s(\mathfrak{F})$ and $\hat{u}_{\mathcal{I}}(\mathfrak{F})$ into Eq(3.8), the overall power function $\hat{\mathcal{P}}(\mathfrak{F})$ can be expressed as:

$$\begin{aligned}\hat{\mathcal{P}}(\mathfrak{F}) &= P_{dyn}^{peak} \cdot \hat{f}_d(\mathfrak{F}) \cdot \hat{u}_{\mathcal{I}}(\mathfrak{F}) + P_{sta}^{peak} \cdot \hat{f}_s(\mathfrak{F}) \\ &= P_{dyn}^{peak}\left(\sum_{i=0}^{4} \hat{\alpha}_i \mathfrak{F}^i\right)\left(\hat{c}_1 \cdot \mathcal{I} + \hat{c}_2\right) + P_{sta}^{peak}\left(\sum_{j=0}^{6} \hat{\alpha}'_j \mathfrak{F}^j\right)\end{aligned} \tag{3.15}$$

We collect the power traces, both dynamic and static power, from McPAT with the settings described in Section 3.3.3. In addition to the nominal frequency (2.66GHz), the frequencies are also set to the range (from 4.95GHz to 1.28GHz) at which McPAT has been extensively validated. Please note that NTC is not included here, since McPat has not been validated for NTC voltage values. Furthermore, only the power values at the nominal frequency are used to train $\hat{u}_{\mathcal{I}}(\mathfrak{F})$ as described in Section 3.3.3. In other words, the power values calculated not at the nominal frequencies are not involved in any part of the training process of $\hat{u}_{\mathcal{I}}(\mathfrak{F})$ (and thus $\hat{\mathcal{P}}(\mathfrak{F})$) – they are "clean" to test the accuracy of $\hat{\mathcal{P}}(\mathfrak{F})$.

Table3.2 shows the overall accuracy of the learned $\hat{\mathcal{P}}(\mathfrak{F})$. On average, the error is 4.37%, which confirms that the learned model can accurately describe the power-performance relationship under different voltage/frequency levels and work variations.

Table 3.2. Error of the learned $\hat{\mathcal{P}}$. The average error is 4.37% (among all frequencies, all benchmarks).

PARSEC Bench.	blckschls.	bdytrck.	canneal	dedup	fluidanim.
$\mathfrak{F}$ = 4.95GHz	2.57%	3.81%	1.86%	1.26%	2.69%
$\mathfrak{F}$ = 2.66GHz	5.31%	7.39%	4.25%	7.44%	5.35%
$\mathfrak{F}$ = 1.28GHz	3.77%	7.72%	1.38%	3.37%	3.87%
PARSEC Bench.	raytrc.	strmclstr.	swaptis.	vips	
$\mathfrak{F}$ = 4.95GHz	4.90%	2.04%	2.30%	6.18%	
$\mathfrak{F}$ = 2.66GHz	5.76%	5.55%	5.58%	8.45%	
$\mathfrak{F}$ = 1.28GHz	4.02%	4.17%	4.00%	10.43%	
SPLASH-2 Bench.	barnes	fft	fmm	lu	ocean
$\mathfrak{F}$ = 4.95GHz	1.48%	2.80%	5.63%	1.57%	1.26%
$\mathfrak{F}$ = 2.66GHz	4.48%	6.14%	6.69%	4.78%	4.85%
$\mathfrak{F}$ = 1.28GHz	3.79%	4.26%	6.41%	3.34%	4.21%
SPLASH-2 Bench.	radiosity	radix	water	**Avg.**	
$\mathfrak{F}$ = 4.95GHz	1.79%	2.56%	2.44%	**2.77%**	
$\mathfrak{F}$ = 2.66GHz	4.54%	7.23%	5.02%	**5.81%**	
$\mathfrak{F}$ = 1.28GHz	3.11%	4.79%	4.51%	**4.54%**	

3.4 Constrained optimization framework

By leveraging the convexity of the learned model $\hat{\mathcal{P}}(\mathfrak{F})$, we convert the following two related problems into convex programs and find the corresponding optimal solutions: (1) minimizing energy consumption under a predefined throughput requirement, and (2) maximizing throughput while satisfying the power constraint.

3.4.1 Energy minimization under throughput constraint

Since the learned $\hat{\mathcal{P}}$ provides an analytical expression that relates the operating frequency and the corresponding power consumption of a processing core under different workload characteristics, we are now in position to answer the following question: for a multi-core system in which each processor is equipped with a wide-operating range, what is the minimum energy required for a predefined throughput constraint? We would like to point out

that, since the purpose of this chapter is to provide a limit study, we do not discretize the continuous function $\hat{\mathcal{P}}$ into certain frequency-voltage pairs, although we do relax this assumption later[4]. In addition, we assume the workload characteristic (*e.g.*, IPC) of each core in every control interval can be obtained via performance counters.

The constrained energy minimization problem of an n-core CMP can be formulated as:

$$\begin{aligned}
\text{Inputs} &: \text{IPC for each core, } \boldsymbol{\mathcal{I}} = (\mathcal{I}_1, \ldots, \mathcal{I}_n)^T \\
\text{Outputs} &: \text{Frequency for each core, } \boldsymbol{\mathfrak{F}} = (\mathfrak{F}_1, \ldots, \mathfrak{F}_n)^T \\
\text{Objective} &: arg\min_{\mathfrak{F}_i} \sum_{i=1}^{n} \hat{\mathcal{P}}(\mathfrak{F}_i) \times \vartheta \\
\text{Subject to} &: \mathcal{T} = \sum_{i=1}^{n} \mathcal{I}_i \cdot \mathfrak{F}_i \geq Perf_{const} \qquad (3.16) \\
&\quad \sum_{i=1}^{n} \hat{\mathcal{P}}(\mathfrak{F}_i) \leq Power_{const} \qquad (3.17) \\
&\quad \mathfrak{F}_{min} \leq \mathfrak{F}_i \leq \mathfrak{F}_{max} \text{ , } \forall i \qquad (3.18)
\end{aligned}$$

The inputs are the IPC of each core ($\mathcal{I}_i$), and the output are the best operating frequency ($\mathfrak{F}_i$). The objective function is to minimize the energy of the CMP, which equals $\sum_{i=1}^{n} p_i \times \vartheta \approx \sum_{i=1}^{n} \hat{\mathcal{P}}(\mathfrak{F}_i) \times \vartheta$. ϑ is a design-dependent value that represents the duration of each DVFS control epoch. Any positive ϑ can be plugged into the proposed optimization framework, and without losing generality ϑ is set to 1 millisecond (ms) in this chapter. Finally, the selected $\mathfrak{F}_i$ must satisfy the following constraints: (1) the total throughput ($\mathcal{T}$) is higher than or equal to the predefined performance constraint, (2) the total power is less than or equal to the predefined power constraint, and (3) each $\mathfrak{F}_i$ must stay within a proper range defined by $\mathfrak{F}_{min}$ and $\mathfrak{F}_{max}$. The values of $\mathfrak{F}_{min}$ and $\mathfrak{F}_{max}$ depend on the operating modes as depicted by Fig.3.1. One possible extension is to accelerate (solving) this optimization

[4]The experimental results of using DVFS with discretized voltage/frequency levels in wide-operating range are provided in Section 3.4.3

program by (1) forcing Eq(3.16) and Eq(3.17) into equality constraints, and (2) reducing the degree in $\hat{\mathcal{P}}(\mathfrak{F}_i)$ into 3 or lower (with losing some accuracy).

To demonstrate this constrained energy minimization is a convex program, we need to show that the objective function and all the constraints are convex.

Lemma 3.4.1.1. *Let $E(\mathfrak{F})$ represent the objective function, $E(\mathfrak{F}) = \sum_{i=1}^{n} \hat{\mathcal{P}}(\mathfrak{F}_i) \times \vartheta$. E is a convex function over $\mathfrak{F}$.*

Proof. E is a function mapping $\mathbb{R}^n$ to $\mathbb{R}$, denoted as $E : \mathbb{R}^n \to \mathbb{R}$. We prove that E is convex by showing its Hessian matrix is positive semi-definite [31]. The Hessian matrix of E can be derived as:

$$\boldsymbol{H} = \frac{\partial^2 E}{\partial \mathfrak{F}_i \partial \mathfrak{F}_j} = \begin{bmatrix} h_{11} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & h_{nn} \end{bmatrix} \tag{3.19}$$

where $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ is the Hessian matrix, and h_{ii} is the i^{th} diagonal entity of $\boldsymbol{H}$ (design-specified constants P_{dyn}^{peak}, P_{sta}^{peak} and ϑ are dropped for conciseness):

$$\begin{aligned} h_{ii} &= \frac{\partial^2 E}{\partial \mathfrak{F}_i^2} = \frac{\partial^2}{\partial \mathfrak{F}_i^2} \sum_{k=1}^{n} \hat{\mathcal{P}}(\mathfrak{F}_k) \\ &= \frac{\partial^2}{\partial \mathfrak{F}_i^2} \sum_{k=1}^{n} \left\{ \hat{f}_d(\mathfrak{F}_k) \hat{u}_{\mathcal{I}}(\mathfrak{F}_k) + \hat{f}_s(\mathfrak{F}_k) \right\} \\ &= \frac{\partial^2}{\partial \mathfrak{F}_i^2} \left\{ \hat{f}_d(\mathfrak{F}_i) \hat{u}_{\mathcal{I}}(\mathfrak{F}_i) + \hat{f}_s(\mathfrak{F}_i) \right\} \\ &= \hat{f}_d'' \hat{u}_{\mathcal{I}} + \hat{f}_d \hat{u}_{\mathcal{I}}'' + 2\hat{f}_d' \hat{u}_{\mathcal{I}}' + \hat{f}_s'' \end{aligned} \tag{3.20}$$

After some algebraic derivations, $\hat{f}_d''$, $\hat{u}_{\mathcal{I}}''$, $\hat{f}_s''$, and $\hat{f}_d'$ are all positive, by plugging in non-negative learned parameters $\hat{\alpha}$, $\hat{d}$ and $\hat{c}$ described in Eq(3.11)(3.10)(3.14). However, $\hat{u}_{\mathcal{I}}'$ is negative, and therefore we cannot algebraically show Eq(3.20) is always non-negative. In this chapter, we plug in all possible values of $\mathfrak{F}$ and $\mathcal{I}$, and $h_{ii} \geq 0$ for all i in all

cases, which means all the eigenvalues of $\boldsymbol{H}$ are non-negative[5]. Therefore, $\boldsymbol{H}$ is positive semi-definite (denoted as $\boldsymbol{H} \succeq 0$), and thereby E is a convex function. ■

Lemma 3.4.1.2. *Let $\mathcal{T}(\mathfrak{F})$ represent the function of throughput $\mathcal{T}$ over $\mathfrak{F}$ in Eq*(3.16)*. Since $\mathcal{T}(\mathfrak{F}) = \sum_{i=1}^{n} \mathcal{I}_i \cdot \mathfrak{F}_i$, $\mathcal{T}(\mathfrak{F})$ is a concave function over $\mathfrak{F}$, and can be converted into a convex function by negating its sign, $-\mathcal{T}(\mathfrak{F})$.*

Proof. We prove $\mathcal{T}(\mathfrak{F})$ is concave by showing the Hessian matrix of $-\mathcal{T}(\mathfrak{F})$ is positive definite. Note that $\mathcal{T}(\mathfrak{F})$ is not linear in $\mathfrak{F}$; when $\mathfrak{F}_i$ is scaled, $\mathcal{I}_i$ changes correspondingly as Eq(3.7) suggests. By plugging in Eq(3.7) into $\mathcal{T}(\mathfrak{F})$, we have:

$$\mathcal{T}(\mathfrak{F}) = \sum_{i=1}^{n} \mathcal{I}_i \cdot \mathfrak{F}_i = \sum_{i=1}^{n} \frac{\mathfrak{F}_i}{\mathcal{C}_i + (\rho_i \cdot \tau_i) \cdot \frac{\mathfrak{F}_i}{\mathfrak{F}_{uc}}} \tag{3.21}$$

where $\mathcal{C}_i$, ρ_i and τ_i are defined as Eq(3.7), and the subscript i means they are the characteristics specifically for core i. By negating $\mathcal{T}(\mathfrak{F})$, we have:

$$\mathcal{T}'(\mathfrak{F}) = -\mathcal{T}(\mathfrak{F}) = -\sum_{i=1}^{n} \mathcal{I}_i \cdot \mathfrak{F}_i = \sum_{i=1}^{n} \frac{-\mathfrak{F}_i}{\mathcal{C}_i + (\rho_i \cdot \tau_i) \cdot \frac{\mathfrak{F}_i}{\mathfrak{F}_{uc}}} \tag{3.22}$$

all symbols are defined as Eq(3.21). Let $\boldsymbol{H}_{\mathcal{T}'}$ be the Hessian matrix of $\mathcal{T}'(\mathfrak{F})$:

$$\boldsymbol{H}_{\mathcal{T}'} = \frac{\partial^2 \mathcal{T}'}{\partial \mathfrak{F}_i \partial \mathfrak{F}_j} = \begin{bmatrix} h_{11} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & h_{nn} \end{bmatrix} \tag{3.23}$$

where $\boldsymbol{H}_{\mathcal{T}'} \in \mathbb{R}^{n \times n}$, and h_{ii} is the i^{th} diagonal entity of $\boldsymbol{H}_{\mathcal{T}'}$. h_{ii} can be expressed as:

$$\begin{aligned} h_{ii} &= \frac{\partial^2 \mathcal{T}'}{\partial \mathfrak{F}_i^2} = \frac{\partial^2}{\partial \mathfrak{F}_i^2} \sum_{k=1}^{n} \frac{-\mathfrak{F}_k}{\mathcal{C}_k + (\rho_k \cdot \tau_k) \cdot \frac{\mathfrak{F}_k}{\mathfrak{F}_{uc}}} \\ &= 2\mathcal{C}_i \cdot \left[\frac{\rho_i \cdot \tau_i}{\mathfrak{F}_{uc}}\right] \cdot \left[\mathcal{C}_i + \frac{\mathfrak{F}_i \cdot (\rho_i \cdot \tau_i)}{\mathfrak{F}_{uc}}\right]^{-3} \\ &> 0 \end{aligned} \tag{3.24}$$

[5]For a diagonal matrix, the entities on the main diagonal are its eigenvalues.

the inequality $h_{ii} > 0$ holds since $\mathcal{C}_i, \rho_i, \tau_i, \mathfrak{F}_{uc}, \mathfrak{F}_i$ are all positive numbers. Therefore, we have $h_{ii} > 0$ for all i, which means all the eigenvalues of $\boldsymbol{H}_{\mathcal{T}'}$ are positive and thus $\boldsymbol{H}_{\mathcal{T}'}$ is positive definite (denoted as $\boldsymbol{H}_{\mathcal{T}'} \succ 0$). We have proved that $\mathcal{T}'(\boldsymbol{\mathfrak{F}})$ is a convex function over $\boldsymbol{\mathfrak{F}}$ and thereby $\mathcal{T}(\boldsymbol{\mathfrak{F}})$ is a concave function over $\boldsymbol{\mathfrak{F}}$. ■

Lemma 3.4.1.3. *The inequality constraints in Eq*(3.18)*,* $\mathfrak{F}_{min} \leq \mathfrak{F}_i \leq \mathfrak{F}_{max}$ $\forall i$*, are convex.*

Proof. Linear functions are both convex and concave [31] and therefore these two inequalities are convex, which directly shows Eq(3.18) is convex. Consequently, Lemma 3.4.1.3 holds. ■

We have shown that the constrained energy minimization of a chip-multiprocessor (CMP) can be converted into a convex program, which can be solved efficiently by Interior Point method and Dual methods [31]. In this chapter, we implement the Interior Point method by using Matlab® to obtain the optimal $\boldsymbol{\mathfrak{F}}$ that minimizes the energy consumption of a CMP while satisfying the throughput and other physical constraints. The Interior Point method is extremely fast, taking only 5ms (in Matlab) to select the best $\boldsymbol{\mathfrak{F}}$ each control epoch and opening the possibility of its applicability in an online setting.

3.4.2 Throughput optimization under power constraint

To best understand the advantages of the wide-range DVFS that includes both NTC and Turbo mode, we also need to explore the optimality from the performance perspective − the maximum throughput under the peak power and other physical constraints. This

constrained throughput optimization problem can be formulated as follows:

$$\begin{aligned}
\text{Inputs} \quad &: \quad \text{IPC for each core, } \boldsymbol{\mathcal{I}} = (\mathcal{I}_1, \ldots, \mathcal{I}_n)^T \\
\text{Outputs} \quad &: \quad \text{Frequency for each core, } \boldsymbol{\mathfrak{F}} = (\mathfrak{F}_1, \ldots, \mathfrak{F}_n)^T \\
\text{Objective} \quad &: \quad arg\max_{\mathfrak{F}_i} \ \mathcal{T}(\boldsymbol{\mathfrak{F}}) \ , \mathcal{T}(\boldsymbol{\mathfrak{F}}) = \sum_{i=1}^{n} \mathcal{I}_i \cdot \mathfrak{F}_i \\
\text{Subject to} \quad &: \quad \sum_{i=1}^{n} \hat{\mathcal{P}}(\mathfrak{F}_i) \leq Power_{const} \\
&\quad \mathfrak{F}_{min} \leq \mathfrak{F}_i \leq \mathfrak{F}_{max} \ , \forall i
\end{aligned}$$

The inputs, outputs and constraints rely on the same metrics as those used in the formulation of energy minimization in Section 3.4.1. The objection function here is to maximize $\mathcal{T}(\boldsymbol{\mathfrak{F}})$, which has been shown to be a concave function in Lemma 3.4.1.2. Furthermore, maximizing a concave function is equivalent to minimizing the corresponding negated function:

$$arg\max_{\mathfrak{F}_i} \ \mathcal{T}(\boldsymbol{\mathfrak{F}}) \ \equiv \ arg\min_{\mathfrak{F}_i} \ -\mathcal{T}(\boldsymbol{\mathfrak{F}}) \tag{3.25}$$

By replacing the original objective function with Eq(3.25), the constrained throughput optimization is converted into a convex program, and the best $\boldsymbol{\mathfrak{F}}$ that leads to the maximum throughput can be calculated efficiently via the methods described in Section 3.4.1.

3.4.3 Impact of discrete V/F levels

So far, to explore the maximum benefit of deploying DVFS within the extended operating range, we assumed the V/F levels to be continuous. An interesting question arises: how much optimality, in terms of energy reduction or throughput improvement, is lost due to the discretization of V/F levels? In other words, we aim to quantify the difference between the theoretically optimal solution assuming continuous V/F levels with the case where only few discrete levels are available. By discretizing the V/F levels, the constrained optimization problems, including both energy minimization and throughput maximization, can be

formulated as an integer linear program (ILP) [31]. In this chapter, the normalized frequencies are discretized into four levels: [1.85, 1, 0.77, 0.059] that represent TB, nominal, low and NTC, respectively. To assess the impact of discrete V/F levels on resulting power savings or performance improvement, Section 3.6 includes for comparison what the theoretical (yet, inefficient, given its NP-hardness) ILP formulation would provide when compared to the approach proposed in this chapter. From a runtime complexity perspective, the ILP formulation can be relatively inefficient, taking several minutes when implemented in Matlab.

3.5 Implementation flow

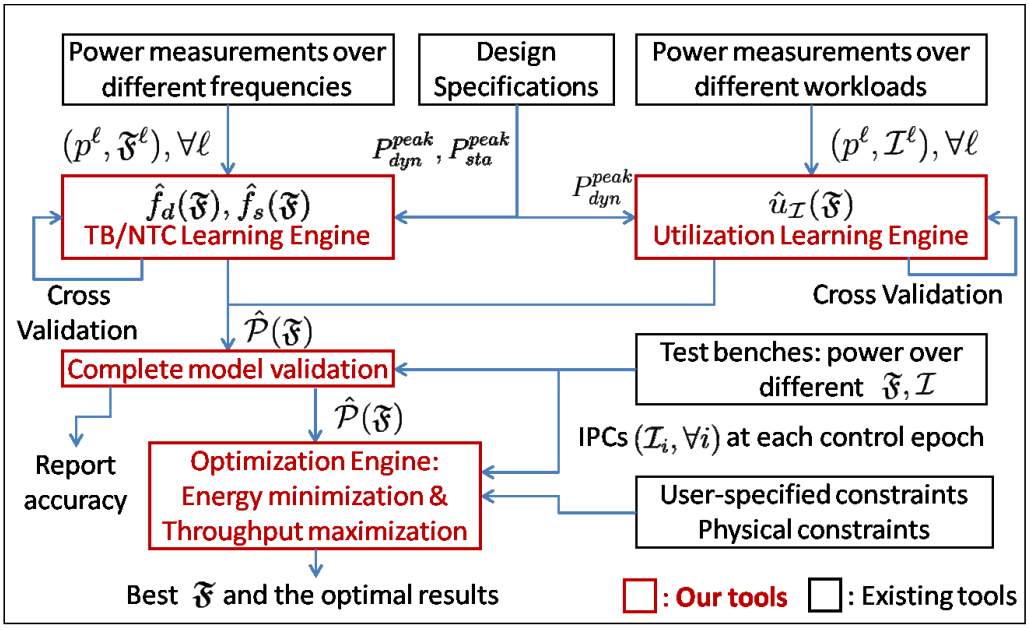


Figure 3.4. Implementation flow

The flowchart is provided in Fig.3.4. To begin with, we collect the dataset $(p^\ell, \mathfrak{F}^\ell)$ for learning $\hat{f}_d(\mathfrak{F})$ and $\hat{f}_s(\mathfrak{F})$ that captures the changes of power consumptions at wide-range operating frequencies (from TB to NTC) as described in Section 3.3.2. Next, we learn $\hat{u}_{\mathcal{I}}(\mathfrak{F})$ that models the dynamic power changes according to the workload variations as described in Section 3.3.3. The design-specific P_{dyn}^{peak} is set to 70% of thermal design power

(TDP)[6], and P_{sta}^{peak} is set to 30% TDP. By plugging $\hat{f}_d(\mathfrak{F})$, $\hat{f}_s(\mathfrak{F})$ and $\hat{u}_{\mathcal{I}}(\mathfrak{F})$ into Eq(3.8), $\hat{\mathcal{P}}$ is obtained. We then validate the learned $\hat{\mathcal{P}}$ with the performance and power values provided by Sniper and McPAT executing PARSEC and SPLASH-2 benchmarks at different operating frequencies, as described in Section 3.3.4. The average error is 4.37%, *i.e.*, 95.63% accuracy is achieved. Furthermore, the overhead of voltage transitions for DVFS is less than 9ns [35] and therefore is neglectable during each control epoch (1ms). We emphasize that the proposed learning framework is generic and is not restricted to a certain simulator or application.

To perform the limit study of the maximum benefits of deploying wide-range operations from TB to NTC, the workload characteristics ($\mathcal{I}$, $\mathcal{C}_{comp}$, ρ and τ in Eq(3.7)) of each processing core along with user-specified constraints are fed into the optimization framework described in Section 3.4.1 and 3.4.2 to select the best $\mathfrak{F}$ for each control epoch. Note that workload characteristics from other sources, such as on-chip performance counters, can also be plugged into the proposed framework for calculating the best $\mathfrak{F}$. Finally, the performance constraint in Eq(3.16) and the power constraint in Eq(3.17) are set to the throughput achieved and power consumed, respectively, under the nominal V_{dd} and $\mathfrak{F}$ for each benchmark.

3.6 Experimental Results

The experimental results of (1) energy minimization under throughput requirements, and (2) throughput maximization under power constraints are provided in the next three subsections, respectively.

3.6.1 Energy analysis

Here, we evaluate the energy minimization under throughput and other physical constraints, as described in Section 3.4.1, by using DVFS over a wide range of operations.

[6]To enable fair comparisons, the new TDP is set to $95/4$W for a processing core, since Gainestown® is a quad-core CMP and its TDP is 95W.

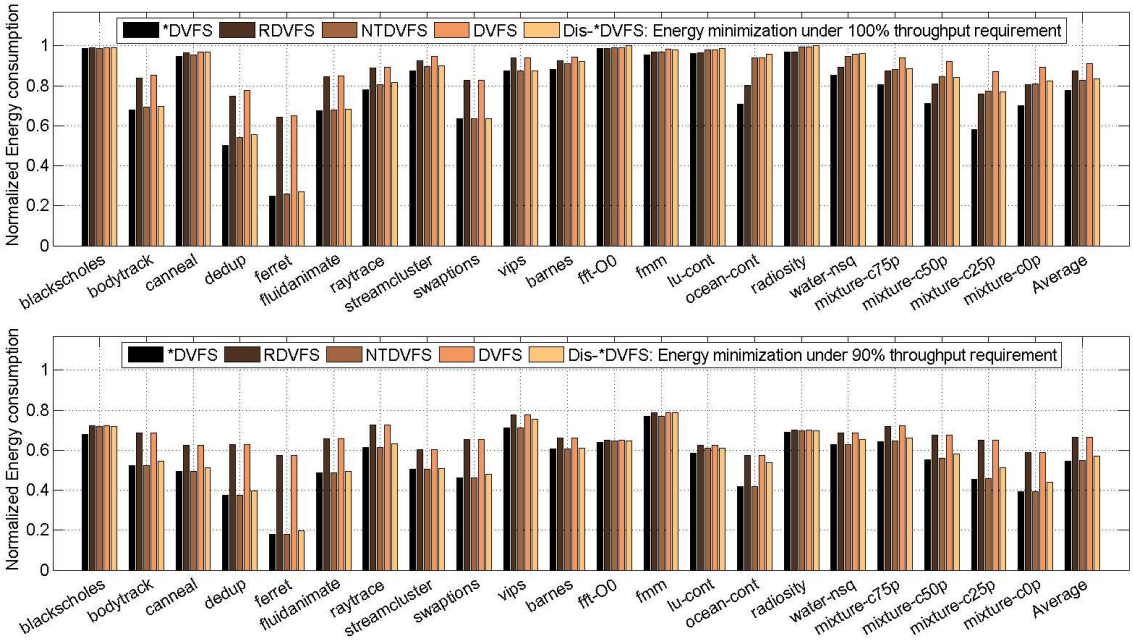


Figure 3.5. Energy minimization under throughput requirements. Top: minimum energy achieved under the 100% throughput constraint. Bottom: minimum energy achieved under the 90% throughput constraint.

Fig.3.5 shows the energy consumption of each PARSEC and SPLASH-2 benchmark and workload mixture. Each workload mixture is created by randomly mixing different PARSEC and SPLASH2 benchmarks to form a heterogeneous multi-threaded application. For example, *mixture_c50p* represents a mixture in which 50% of threads are core-bound (computationally-intensive)[7] and the rest of them are memory-bound (memory-access-intensive) [22].

Fig.3.5(top) shows the minimum energy achieved under the 100% throughput (TP) constraint (or iso-performance), and Fig.3.5(bottom) shows the minimum energy achieved under the 90% TP constraint, which means at most 10% of TP is sacrificed in order to reduce energy consumption. Note that all results are **optimal** and all constraints listed in Eq(3.16)(3.17)(3.18) are met. "*DVFS" stands for applying DVFS within the extended operating range that includes TB, nominal and NTC, "R-DVFS" corresponds to DVFS with TB capabilities, "NT-DVFS" corresponds to DVFS with NTC capabilities, "DVFS" in-

[7] A core-bound thread usually requires a smaller working set of caches, and therefore most operations can be done without intensive DRAM accesses [22].

cludes regular (nominal) DVFS only, and "Dis-*DVFS" represents *DVFS with discretized V/F levels based on an ILP formulation as described in Section 3.4.3. All results are normalized to the energy consumption at the operating frequency at nominal $V_{dd} = 0.7$V (denoted as $\mathfrak{F}_{nom}$) without DVFS.

From Fig.3.5(top), we have the three important observations. (1) *DVFS achieves a significant energy reduction of 22.33% without compromising any throughput, of which 13.28% comes from using extended range DVFS with TB and NTC whereas 9.02% is from using DVFS with conventional operating range. This directly demonstrates the importance and the advantage of applying DVFS over a wide-operating range. We also extend the operating range to include sub-threshold, and the energy reduction is only 0.8% (23.2% overall reduction) better. (2) NT-DVFS, R-DVFS, DVFS and Dis-*DVFS reduce energy consumption by 17.29%, 12.52%, 9.02% and 16.55%, respectively. Compared to DVFS, adding Turbo mode (R-DVFS) helps energy reduction, which has also been pointed out in previous literature [36]. Also, discretizing V/F levels leads to 5.78% (22.33%−16.55%) loss in optimality on average, even if only four discrete levels are available. (3) Heterogeneous workloads (workloads with different thread-level characteristics) receive more benefits from *DVFS. For example, compared to *blackscholes* in which each thread has similar characteristics, the energy reduction of *ferret* is more significant. *ferret* is implemented using the "pipeline programming model" [22] that has distinct characteristics for each thread and frequent thread migration. These two facts make its behavior similar with heterogeneous workloads; the heterogeneity among threads provides more room for leveraging performance and energy consumption of the whole CMP via the proposed framework, and therefore a more significant energy reduction can be achieved without compromising the overall throughput. A similar phenomenon can be observed for *dedup* and *mixture_c25p*.

Fig.3.5(bottom) shows even more significant energy reduction: on average 45.29% of energy is reduced by using *DVFS with at most 10% throughout reduction, of which 11.85% can be attributed to extended range DVFS only. Also from Fig.3.5(bottom), we can ob-

serve that NT-DVFS performs almost as well as *DVFS, whereas R-DVFS behaves similarly with DVFS. Dis-*DVFS reduces the average energy by 43%. In this case, only 2.29% (45.29%−43%) energy saving is lost compared to optimal results due to V/F discretization. All the above evaluations are enabled exclusively by our proposed learning-based optimization framework.

3.6.2 Performance analysis

In this section, we evaluate the throughput improvement under power constraints as described in Section 3.4.2. Fig.3.6 shows the throughput of each PARSEC and SPLASH-2 benchmark and workload mixture. The labels are as defined in Section 3.6.1. All results are normalized to the throughput at the $\mathfrak{F}_{nom}$ without DVFS.

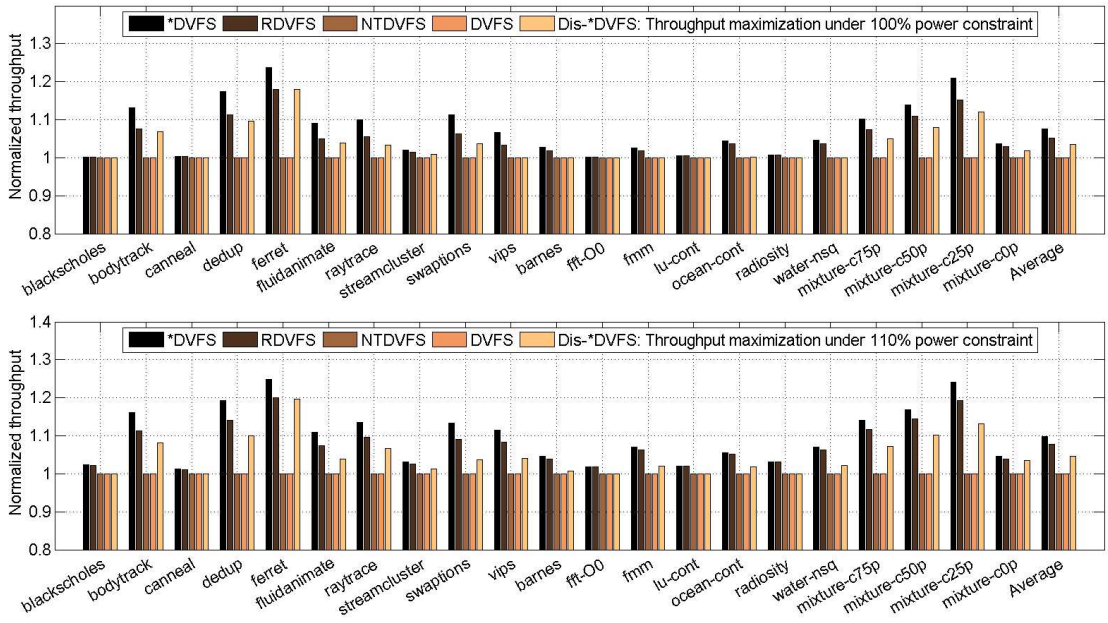


Figure 3.6. Throughput improvement under power constraints. Top: Throughput improvement under 100% power constraint. Bottom: Throughput improvement under 110% power constraint.

From Fig.3.6(top), under the iso-power condition, *DVFS, R-DVFS and Dis-*DVFS achieve 7.54%, 5.11% and 3.48% average throughput improvement, respectively. This fact

again necessitates the use of DVFS over a wide-operating range that includes TB, nominal and NTC, since doing so helps improve the performance. Note that NT-DVFS and DVFS are not allowed to select the frequency higher than $\mathfrak{F}_{nom}$, and therefore their throughput is the same as the baseline. Furthermore, benchmarks with more thread-level heterogeneity receive higher improvement. For example, *dedup* and *ferret* have 17.35% and 23.83% higher throughput, respectively, whereas *blackscholes* only improves by 0.23%.

Fig.3.6(bottom) shows the throughput improvement under 110% of power constraint. The improvement for each benchmark is similar with the results in Fig.3.6(top). On average, *DVFS achieves 9.86% throughput improvement under the 110% power constraint. In this case, Dis-*DVFS improves throughput by 4.68% on average, which is approximately 5% away from optimal performance improvement. If more discrete levels are available, the loss in optimality will be lower. Again, all the results are theoretically-**optimal**, which directly demonstrates the strength of the proposed learning-based optimization framework.

3.6.3 Frequency selection analysis

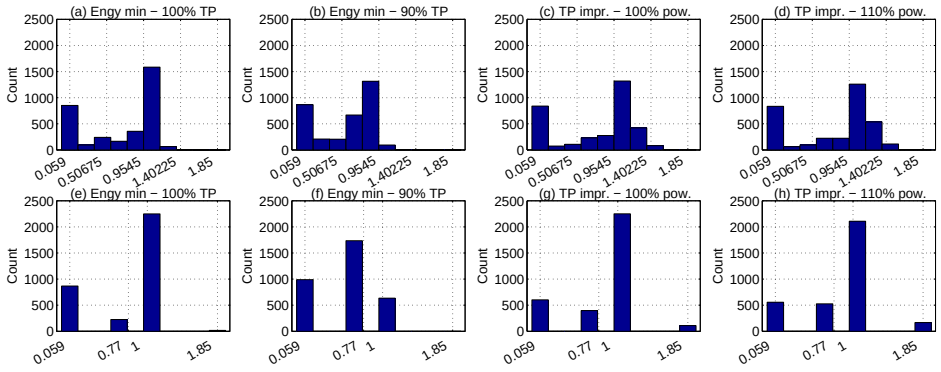


Figure 3.7. The Histogram of the optimal $\mathfrak{F}$ usage. (a)-(d) Continuous frequency selected by *DVFS. (e)-(h) Discretized frequency selected by Dis-*DVFS. X-axis represents the normalized value of $\mathfrak{F}$ and Y-axis stands for the counts.

Finally, we examine the optimal $\mathfrak{F}$ usage for energy minimization and throughput improvement, *i.e.*, the proportion of a certain value of $\mathfrak{F}$ is selected by *DVFS and Dis-*DVFS

to minimize the energy or improve the throughput during each control epoch. The normalized frequency ranging from [1.85, 0.059] (the maximum $\mathfrak{F}$ in TB and the minimum $\mathfrak{F}$ in NTC) is equally discretized into 10 bins. Let us first focus on the frequency selected for energy minimization. Fig.3.7(a)(e) demonstrates the respective $\mathfrak{F}$ selected by *DVFS and Dis-*DVFS under 100% throughput requirement, and Fig.3.7(b)(f) shows the results under 90% throughput requirement. As Fig.3.7(a)(e) shows, both *DVFS and Dis-*DVFS are forced to select a $\mathfrak{F}$ (close to $\mathfrak{F}_{nom}$) to satisfy the 100% throughput requirement, whereas in Fig.3.7(b)(f), we can observe that both *DVFS and Dis-*DVFS tend to select a lower $\mathfrak{F}$ to reduce energy since the constraint allows at most 10% loss in throughput.

Fig.3.7(c)(g) demonstrates the respective $\mathfrak{F}$ selected by *DVFS and Dis-*DVFS under 100% power constraint, and Fig.3.7(d)(h) shows the results under 110% power constraint. As Fig.3.7(c)(d) shows, *DVFS takes advantage of continuous frequency levels and selects values slightly higher than the nominal frequency ($\mathfrak{F}_{nom}$) to improve throughput while satisfying the power constraint at the same time. On the other hand, most of the $\mathfrak{F}$ selected by Dis-*DVFS are one ($\mathfrak{F}_{nom}$); very rarely $\mathfrak{F} = 1.85$ (TB) is selected. The results here suggest an opportunity of re-discretizing V/F levels without losing much optimality, which we leave as a future work.

3.7 Discussion

In this chapter, we adapt the model-selecting technique and LOOCV from machine learning to learn the best constrained-posynomial $\hat{\mathcal{P}}$ for modeling the workload-dependent power-frequency relationship over an extended range. Based on the convexity provided by the learned $\hat{\mathcal{P}}$, two optimization frameworks are proposed: energy minimization under throughput constraints and throughput maximization under power constraints. The experimental results confirm the benefits of the wide-operating range: on average, additional 13.28% of energy is reduced under the iso-performance condition and the average throughput is increased by additional 7.5% under the iso-power condition, compared with DVFS

in the nominal operating range. One straightforward direction of future work is to extend these evaluations to include process variations which are known to become increasingly important in NTC.

Chapter 4

Joint optimization of power and performance for multi-core systems

Moving from a multi-core system that contains only processing cores (the focus of Chapter 3) to a system that incorporates both processing cores and on-chip communication fabrics, a new control paradigm is needed to further improve the performance as well as the energy efficiency. In this chapter, we apply reinforcement learning (RL) to determine the operating points for both processing cores and on-chip communication fabrics. Specifically, the parameters used during the reinforcement learning are learned via the Bayesian inference.

4.1 Network-on-chip for multi-core systems

Over the last decade, microprocessor design trends have shifted to chip-multicores from the classic monolithic, single core systems. In a multi-core system, processing elements or cores must communicate with each other under parallel, multi-threaded workloads, thereby potentially creating performance bottlenecks in the communication fabric [37]. To reduce this overhead, the network-on-chip (NoC)[1] paradigm [37, 38] has been proposed as a promising solution for on-chip communication for massively-integrated CMPs. However,

[1]Network-on-chip (NoC) and on-chip networks will be used interchangeably throughout this chapter.

the enhanced performance and capabilities of such platforms are usually constrained by the on-chip power consumption. While dynamic power management has been extensively studied for multi-core systems in the context of core-only or uncore (on-chip communication fabric) only, the cooperative power management of core and uncore[2] has remained a critical issue not sufficiently explored.

In the context of using dynamic voltage and frequency scaling (DVFS) for minimum power consumption under performance constraints, a possible cooperative power management for both core and uncore resources introduces additional challenges that require maintaining appropriate performance levels for parallel applications executing on the system. For example, in multi-threaded applications, spin locks and other synchronization mechanism may amplify small timing differences into very different program execution paths [39], thereby impacting the memory system behaviors substantially. In addition, a significant mismatch between core and uncore frequency may cause unexpected traffic contentions and therefore, may result in significant performance penalty [40]. As a result, how to reliably control cores and uncores in synergy via DVFS while maintaining power constraints remains an open question that needs to be addressed in the context of advanced multi-core systems.

4.1.1 Contributions and chapter organization

To the best of our knowledge, our learning model brings the following novel contributions:

- We present, for the first time, the evaluation of and comparison among core-only, uncore-only and *cooperative core/uncore DVFS control for performance boosting*. The experimental results confirm that performing DVFS for cores or uncores separately may not be effective for NoC-based CMPs as they are characterized by smaller performance per unit of energy gains.

[2]In this chapter we refer to uncore resource as representing the communication fabric only, such as routers and links.

- Compared to conventional DVFS schemes that address power reduction under performance constraints, we propose a *"reverse"* DVFS: maximize performance while ensuring that power stays within prescribed power constraints.

- We evaluate the proposed RL-based, cooperative DVFS control for both cores and uncores with a wide spectrum of parallel, multi-threaded applications. The experimental results confirm the effectiveness − on average 10.9% of program execution time is reduced while satisfying given power constraints.

This chapter is organized as follows. Section 4.2 provides the target architecture and detailed formulation. Section 4.3 elaborates on reverse DVFS (RDVFS) by using reinforcement learning. Section 4.6 explains the implementation flow, Section 4.7 and Section 4.8 provide the experimental results and discussions, respectively.

4.2 Problem formulation

We introduce the architecture and the problem formulation used herein. The architectural configuration described is to facilitate the explanation of the proposed methodology, and the results are generalizable for any multi-core system other than the one described in this session.

4.2.1 Target Architecture

The architecture used throughout this chapter is a symmetric, NoC-based CMP that consists of 16 tiles, and each tile contains a Pentium4® core, a private L1 cache, a shared L2 cache and an on-chip router. Table 4.1 provides the detailed architectural parameters. These 16 tiles are placed in a 4×4 mesh manner. A flit-based mechanism is used for the NoC architecture. The router design follows the standard 5-stage pipeline [41]. Furthermore, each processing core and its corresponding on-chip router are assumed equipped with Intel's Turbo Boost® technology [42]. Therefore, the voltage and frequency pairs,

Table 4.1. Architectural parameters.

Core Parameters	Values	Uncore Parameters	Values
Number of cores	16	Number of routers	16
Core model	Pentium 4®	Nominal frequency	3.0 GHz
Nominal frequency	3.0 GHz	Router pipeline stages	5 stages
L1-I/D caches	Private 64KB, 8-way SA, LRU	Flit size	16 Bytes
L2 caches	Shared 4MB, 32-way SA, LRU	Number of virtual channels	4 per port
Cache coherence	MOESI protocol [43]	Buffer size	4 Bytes
DRAM	2 GB	Network topology	4×4 mesh
Technology	45nm node with V_{dd} =1.0V	Routing algorithm	X-Y routing

Table 4.2. Voltage and frequency pairs.

DVFS V/F Pairs		Values	DVFS V/F Pairs		Values
Turbo	V_{dd}	1.3V	Baseline	V_{dd}	1.0V
	Freq	3.75GHz	(Nominal)	Freq	3.0GHz
Low	V_{dd}	0.8V	Very Low	V_{dd}	0.65V
	Freq	2.35GHz		Freq	2.0GHz

denoted as V/F pairs, can be set to (1) turbo: 1.3V, 3.75GHz, (2) baseline (nominal): 1.0V, 3GHz, (3) low: 0.8V, 2.35GHz and (4) very low: 0.65V, 2.0GHz. These V/F pairs are listed in Table 4.2. For conciseness, in the remainder of this chapter, when we mention the change of frequency, we actually refer to the change of both voltage and frequency, *i.e.*, V/F pairs. In order to compare with conventional core-only or uncore-only DVFS, we assume that each core and router can be set to different frequencies to best explore the advantages of the cooperative DVFS control.

4.2.2 Detailed formulation

The formulation of the cooperative core/uncore DVFS is described as follows. First, the following two inputs are given: (1) a NoC-based CMP, and (2) a parallel, multi-threaded workload that can be executed on the target CMP. The decision variables here are the frequency of each core and router. Under initial conditions, all frequencies are set to the baseline value (3GHz). The objective function to be maximized is overall performance. In this chapter, the throughput of the CMP is used as the performance metric. Generally,

performance can be expressed as a function of the frequency and the machine states S^t (the state at the t^{th} control epoch). Finally, the power consumption of the whole CMP, including both cores and uncores, must be less than or equal to the power constraint (Pow_{const}) at all times.

4.3 Reverse dynamic voltage and frequency scaling (RDVFS) via reinforcement learning

This section elaborates on two main components of the proposed reinforcement learning (RL) framework: *Markov decision process* (MDP) and *maximum a posteriori* (MAP) estimate.

4.4 Markov decision process (MDP)

The concept of RL can be described via the interactions between an agent and the environment with uncertainties. The agent attempts to find the best action on the fly for interacting with the various states of the environment, in order to receive the highest reward. Therefore, a MDP-based learning model, such as V learning or Q learning [14], consists of: (1) an agent; (2) a finite state space $S \in \{s_1 \dots s_n\}$; (3) a set of available actions $A \in \{a_1 \dots a_m\}$; and (4) a reward function $r^t = r(S^t, A^t)$ where t represents the time. As a convention in statistics, capital variables such as S represent random variables and lowercase variables such as s_1 stand for the values observed. The goal of the agent is to maximize its expected long-term reward. This can be achieved by learning a policy π which can be viewed as a mapping between the states and the actions. Table 4.3 lists the parallel between MDP and dynamic power management concepts.

Figure 4.1 presents a MDP model. At each time point, the agent chooses an action $a \in A$ based on the state S^t to receive the long-term highest reward. Based on the current state S^t

Table 4.3. MDP and dynamic power management.

MDP	Dynamic power management
Agent	Controller
Select an action	Select a Voltage/Frequency
Environment states	Machine states
Rewards	High performance

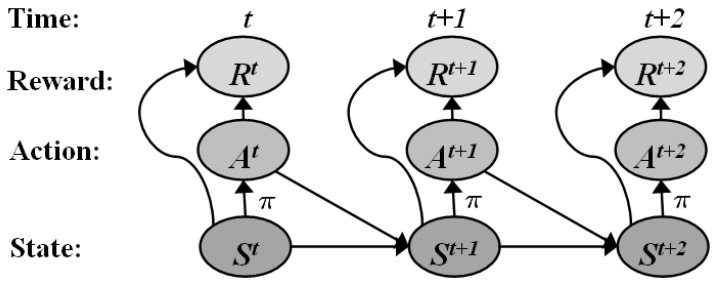


Figure 4.1. A Markov decision process.

and the action A^t , the state transition probability of S^{t+1} can be calculated as:

$$\mathbb{P}(S^{t+1} = s_i | S^t = s_j, A^t = a_k) = \theta_i \tag{4.1}$$

where $s_i, s_j \in S$ and $a_k \in A$. Here, we define this transition probability as θ_i. More precisely, θ_i represents the probability that state s_i will be reached given that s_j, a_k was reached during the previous time point. Given s_j, a_k, let $\theta = \{\theta_1, \cdots, \theta_{|S|}\}$ represent the probability for each value taken by S^{t+1}, *i.e.*, $\{s_1, \cdots, s_{|S|}\}$. For simplicity of explanation, we assume that the states of MDP in Figure 4.1 are observable instead of hidden. In other words, on-chip performance or power counters are readily available for all resources; if the states are hidden, expectation maximization (EM) [14] or other algorithms can be used to predict the state. Furthermore, the first-order Markov assumption is used here: the probability of S^{t+1} depends on only S^t and A^t, and therefore $\mathbb{P}(S^{t+1}|S^t, A^t, S^{t-1}, \dots) = \mathbb{P}(S^{t+1}|S^t, A^t)$. No information before time t is required to calculate the conditional probability.

The new state S^{t+1} provides a reward to the agent, and the agent learns the control policy $\pi : S \rightarrow A$ to maximize the long-term expected reward $\sum_{t=0}^{\infty} \gamma^t \mathbb{E}[r^t]$, where γ

is the discount factor between (0,1). γ is used to discount the future reward so that the long-term reward will converge to a certain value after a sufficiently long time. Since the state transition is not deterministic, the expected value is calculated as the expected reward: $\mathbb{E}_\theta[r^t] = \sum_{s_i \in S} r^t \times \theta_i$. Given $\pi : S \rightarrow A$, we can define $V^\pi(s)$ as $\sum_{t=0}^{t=\infty} \gamma^t \mathbb{E}_\theta[r^t]$, where $V^\pi(s)$ represents the long-term expected reward an agent can receive if the agent follows the action sequence chosen according to π, starting at state s. Then, the best policy is $\pi^* = \arg\max_\pi V^\pi(s), \forall s$. Assuming that the state transition probability θ is known, we can calculate the best π^* and the $V^\pi(s)$ in a recursive way to obtain the best control policy $V^{\pi*}(s)$. If θ remains unknown, the learning problem relies on the estimates of the state transition probability. In this chapter, we will demonstrate that, by using *maximum a posteriori* (MAP) estimate with a proper prior distribution, learning θ can be much faster.

4.5 MAP estimate for state-transition probability

The key step in MDP is to learn the unknown state transition probability, $\theta_i = \mathbb{P}(S^{t+1} = s_i | S^t = s_j, A^t = a_k)$, as mentioned in Section 4.4. θ heavily depends on the characteristics of the application workload and the underlying processor design. In other words, θ is both machine- and application-dependent. Therefore, θ has to be learned on the fly during the program execution, which is known as the main strength of RL $-$ adaptivity.

Here, we applied MAP to estimate the values of θ. Like mentioned in Section 2.2.2, MAP is closely related to maximum likelihood estimate (MLE), but it employs an augmented optimization which incorporates a *prior* distribution over θ. First, we define d as the observed data of $S^{t+1} = s^{t+1}$ given $(S^t = s_j, A^t = a_k)$, and we want to estimate θ, denoted by $\hat{\theta}$ which equals the maximum value of $\mathbb{P}(\theta|d)$ (or called mode in statistics). $\mathbb{P}(\theta|d)$ can be interpreted as: after d happened, how should the probability of θ be updated? Recall the Bayes rule in Eq(2.7), $\mathbb{P}(\theta|d)$ can be rewritten as:

$$\mathbb{P}(\theta|d) \propto \mathbb{P}(d|\theta) \times \mathbb{P}(\theta) \tag{4.2}$$

where $\mathbb{P}(\theta|d)$ is the posterior distribution, $\mathbb{P}(d|\theta)$ represents the data likelihood and $\mathbb{P}(\theta)$ is the prior distribution. The normalization term $1/\mathbb{P}(d)$ is not shown here since it is a constant and does not affect estimates. Once we determine the prior distribution $\mathbb{P}(\theta)$, the posterior distribution $\mathbb{P}(\theta|d)$ can be updated by Eq(4.2) as the data likelihood $\mathbb{P}(d|\theta)$ changes since more state transitions are observed by the agent (or the controller). The likelihood function $\mathbb{P}(d|\theta)$ follows the multinomial distribution:

$$\mathbb{P}(d|\theta) = \frac{N!}{\prod_{i=1}^{i=|S|} x_i!} \prod_{i=1}^{|S|} \theta_i^{x_i} \tag{4.3}$$

where $x_i \in \{0, \cdots, N\}$ and $\sum x_i = N$. x_i represents the occurrences of $S^{t+1} = s_i$, given $S^t = s_j, A^t = a_k$. N represents the total number of occurrences of $S^t = s_j, A^t = a_k$. The numerical intuition behind this multinomial likelihood is that, given N, we could calculate how likely S^{t+1} will take on the value of s_i by using its corresponding occurrence x_i. With Eq(4.3), the data likelihood can be calculated, but we still need to determine the prior distribution to calculate the posterior distribution.

The prior distribution is usually selected based on one's domain knowledge. By selecting a good prior, the posterior distribution can converge faster, and thus the estimate of state transition probability, $\hat{\theta}$, can be obtained earlier during the program execution. In this chapter, we propose to use a Dirichlet distribution as the prior distribution:

$$\mathbb{P}(\theta) = \frac{\prod_{i=1}^{i=|S|} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{i=|S|} \alpha_i)} \prod_{i=1}^{|S|} \theta_i^{\alpha_i - 1} \tag{4.4}$$

where α_i represents the "hallucinated[3] counts of s_i, and $\Gamma(\alpha_i)$ is the Gamma function that equals the factorials of $(\alpha_i - 1)$. α_i is similar to x_i in Eq(4.3) except that x_i represents the actual counts of s_i , whereas α_i stands for our "belief" or "domain knowledge" of how

[3]These hallucinated counts α_i can be any positive integer or 0. If the learning process is sufficiently long, θ will be learned correctly and α_i is irrelevant. This is known as "the prior is forgotten [14]"

many times s_i should happen before any data are given. Empirically, the range of α_i is set to $10^1 - 10^2$.

Next, we multiply Eq(4.3) by Eq(4.4) to obtain the posterior distribution:

$$\begin{aligned} \mathbb{P}(\theta|d = \{x_1 \cdots x_{|S|}\}) &= \frac{\prod_{i=1}^{i=|S|} \Gamma(\alpha_i + x_i)}{\Gamma(N + \sum_{i=1}^{i=|S|} \alpha_i)} \prod_{i=1}^{|S|} \theta_i^{\alpha_i + x_i - 1} \\ &= \frac{\prod_{i=1}^{i=|S|} \Gamma(\beta_i)}{\Gamma(\sum_{i=1}^{i=|S|} \beta_i)} \prod_{i=1}^{|S|} \theta_i^{\beta_i - 1} \end{aligned} \tag{4.5}$$

where $\beta_i = \alpha_i + x_i$. The numerical intuition of β_i is that, besides the actual observations (x_i), we add the hallucinated counts (α_i) to adjust $\mathbb{P}(\theta|d)$. In other words, intuitively the domain knowledge is used to aid the estimate of the state transition probabilities. Furthermore, it can be seen that Eq(4.5) and Eq(4.4) are in the same general form of Dirichlet distributions (only the parameters α, β are different), which is known as *"conjugate"* prior and posterior pairs. Each time a new state transition d is observed, the posterior distribution can be updated by the multiplication of new data likelihood and the prior by using Eq(4.2), and thus, the updated posterior can be fed back into Eq(4.2) as an *updated prior* ready to estimate the new posterior since they are in the same form.

4.6 Implementation flow

In this section, we describe the experimental setup and the simulation infrastructure in detail.

4.6.1 Performance modeling infrastructure

For the simulation infrastructure, we use Simics [44] and GEMS [5] as a full-system, many-core simulator to evaluate the proposed RL control. The operating system is configured as Linux in Simics. Furthermore, RUBY [5] and GARNET [41] are embedded in GEMS to enable the functional and timing simulations of the cache system and the on-chip

communication fabrics, respectively. Detailed parameters are mentioned in Section 4.2.1. Default values are used as system parameters if not specifically mentioned. For the workloads considered, we use PARSEC [45] as multi-threaded, parallel applications to evaluate the benefits of the proposed control strategy. PARSEC covers a wide spectrum of data sharing and synchronization, which creates both on-chip and off-chip communication close to realistic workloads.

4.6.2 Power modeling for on-chip resources

We use the power model proposed by [46] to calculate the power consumptions of processing cores. This power model is calibrated by Intel® Xeon® X7350, and the difference between the actual power consumption and the fitted one from [46] is less than 10%. Therefore, it can accurately calculate the power consumption of NoC-based CMPs emulated by our simulation infrastructure.

For the power model of NoC, we use Orion [47] built in GARNET to provide the power consumption of the on-chip communication fabrics, including routers and links. We further modify and shrink the technology node in Orion to 45nm by using the parameters provided by Orion 2.0 [48]. Finally, CACTI [49] is integrated into RUBY to calculate the power dissipated by L1 and L2 caches in different states (*e.g.*, Read, Write and Standby).

4.6.3 Controller design

The proposed RL-based controller can be implemented as a kernel thread executing on each processing core. Also, the proposed MAP can be implemented very efficiently with a table-lookup technique. We implemented and optimized the controller in C++ and evaluated it on our full-system simulator. The timing overhead is around 0.03 ms, while each control epoch (or period) is 1.2ms (therefore, controller overhead is around 2%). The power

overhead of the proposed control is less than 0.32 Watts per tile, which is less than 1% of the thermal design power (TDP)[4].

For the overhead in memory usage, extra memory space is needed to record all state transition probability. Conventionally, the machine states are defined by three performance counters − instruction per cycle (IPC), misses per kilo instruction (MPKI) and router buffer utilization (BU). These three metrics represent the respective metric of performance from cores, caches and on-chip networks, and thus are sufficiently representative as the machine states of the whole CMP. The number of states per tile is $|S| = |IPC| \times |MPKI| \times |BU|$. Here, we use four states to represent $|IPC|$, three states for $|MPKI|$ and two states for $|BU|$, all based on the utilization rates from [50, 51]. Hence, the total number of states per tile is 24 and thus the overhead in memory space is very low.

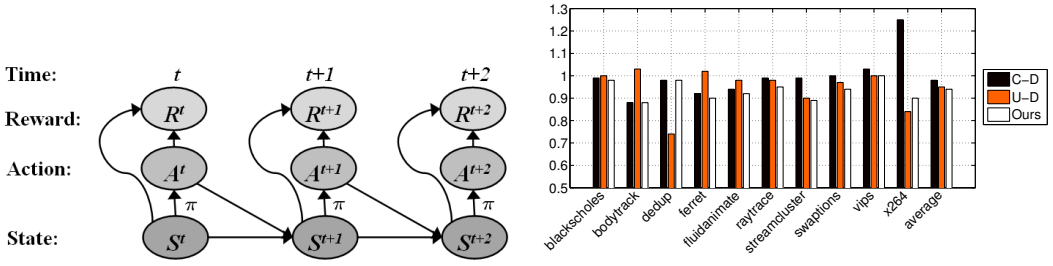
As a comparison, we re-implement the DVFS controls proposed by [50] and [51]. We select *"Threshold-based"* control of [50] and aggressively upscale the frequency settings for improving the performance. The original baseline frequency in [50] is raised to the Turbo mode (3.75GHz), and the rest of settings are changed in a similar fashion. For uncore-DVFS, the *"freqboost"* of [51] is used to maximize the performance − the frequency is set to 3.75GHz (Turbo) by default. The frequency of a router will be reduced if the occupancy of its downstream routers' buffers is greater than 60% [51], which is considered as potential network congestion. Finally, the power constraint is set to the larger value of the peak power consumption from core-only DVFS or uncore-only DVFS, which is around 350Watts to 470 Watts.

4.7 Experimental results

This section presents the experimental results, including the overall analysis, energy-delay product and network latency.

[4]Since the power model [46] is originally for quad-core (Xeon® X7350, TDP=130Watts) and we apply it on the 16-core CMP, the new TDP will be $130 \times 4 = 512$ Watts.

4.7.1 Execution time analysis



(a) Normalized execution time (b) Normalized energy-delay product

Figure 4.2. Overall analysis.

Figure 4.2(a) illustrates the analysis of the program execution time for each PARSEC benchmark. "C-D" stands for the Core-only DVFS based on [50], whereas "U-D" is the Uncore-only DVFS based on [51]. All results are normalized to the "Baseline", *i.e.*, the case without any DVFS. On average, the proposed RL control achieves 10.9% reduction of the execution time, and outperforms C-D and U-D almost in all cases except *Dedup*. *Dedup* is implemented using a "pipeline programming model" [45] that has two properties: (1) distinct characteristics for each thread, and (2) frequent thread migration: every few thousand cycles. These two facts make the state transition probabilities hard to estimate. However, the proposed control still reduces the execution time by around 7.5%. In most core-bound applications, such as *blackscholes*, both C-D and the proposed control approach outperform U-D. On the other hand, in memory-bound applications, such as *X264*, both U-D and the proposed control have better reduction than C-D. It is also worth mentioning that C-D has longer execution time in *StreamCluster* and *X264*, which will be explained later in Section 4.7.2. These results clearly demonstrate the effectiveness and robustness of the proposed control, compared to C-D and U-D.

Figure 4.2(b) provides the comparisons of energy-delay products. It can be seen that the proposed control achieves the lowest energy-delay product on average $-$ 6% reduction compared to the baseline, whereas C-D and U-D achieve only 2% and 4.7% reduction, respectively. This means the energy efficiency of the proposed control policy is higher

than both C-D and U-D. Note that the power constraint is met at all times and for each benchmark. Furthermore, the peak temperatures (calculated by the thermal model proposed by [46]) of all three control policies stay within 60° C for each benchmark, and thus no thermal emergency occurs due to the "*reverse*" DVFS.

4.7.2 Network latency analysis

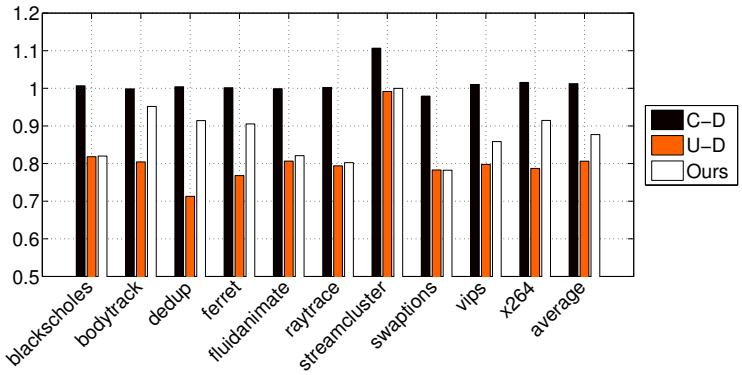


Figure 4.3. Normalized end-to-end network latency.

Here, we examine the network latency under four different control schemes. Figure 4.3 presents the average end-to-end network latency for each benchmark. As it can be seen, U-D effectively reduces the network latency by around 19.1%. Considering the reduction of execution time shown in Figure 4.2(a), it can be seen that the shorter network latency cannot be directly reflected on the shorter execution time in most cases. This fact reveals that U-D, *i.e.*, uncore-only DVFS can be very effective, but only in certain cases. On the other hand, even if the router frequency is fixed at the nominal value, we can see that C-D has longer network latency in *StreamCluster*, *Vips* and *X264*. This is because the uncore-agnostic DVFS may change the packet injection rate in the network, which results in unexpected congestion and thus longer latency. This also explains why we observe that C-D has longer execution time in *StreamCluster* and *X264* in the previous section. In most cases, the network latency of the proposed control is longer than the latency of U-D (sometimes it is a tie) and shorter than the latency of C-D. This is because the proposed control needs to strike

a balance to distribute the power budget to cores or routers for boosting up the frequency, in order to achieve the best performance increase. For example, compared to C-D, the proposed control boosts up the router frequencies (and hence achieves a shorter latency) in *X264*, which in turn reduces the program execution time. All the above results demonstrate that, instead of core-only or uncore-only DVFS, a cooperative control of both cores and uncores is more effective and robust for NoC-based CMPs, which also demonstrates the strength of the proposed RL control.

4.8 Discussion

In this chapter, we evaluate and present the advantages of controlling cores and uncores in synergy for NoC-based CMPs. As pointed out by [40, 10], performing core-solely DVFS without being aware of on-chip traffic may reduce the system performance due to unexpected network congestion. Therefore, we propose a control mechanism based on semi-supervised RL that is highly adaptive and scalable for advanced CMPs. The proposed control mechanism learns the state transition probability on the fly, and selects the best V/F pair to improve the performance. Experimental results show that the proposed control achieves 11% reduction on the program execution time. One possible extension to reduce the design overhead is to adapt the proposed semi-supervised RL on the CMP equipped with a unified V/F island for the whole on-chip communication fabric.

Chapter 5

Thermal modeling for multi-core systems

The demand of high performance inevitably increases the power consumption, which in turn generates heat and elevates the operating temperature of a multi-core system. Excessively high operating temperature damages the reliability of a system, both temporarily (*e.g.*, soft errors) or permanently (*e.g.*, thermal runaway). In this chapter, we elaborate on the thermal modeling via Gray-Box computing: a linear regression with an L1 norm regularizer. Based on the learned model, further thermal optimization (hotspot mitigation) is also performed and demonstrated.

5.1 Thermal issues and reliability

In recent years, thermal issues have severely hindered the development of highly advanced and reliable chip-multiprocessors (CMPs). Excessively high operating temperature is the root of many reliability issues, since the rates of many failure mechanisms will increase exponentially with operating temperature [2]. Also, high operating temperature is known for increasing CMP's power consumption, especially leakage power [2]. The increase of leakage power contributes to the increase of total power consumption, which in

turn increases the operating temperature. This thermal-leakage positive feedback loop may lead to thermal runaway, which in the worst case may burn the chip [9].

Although thermal RC simulation or finite-difference method (FDM) [52] used by prior arts usually guarantee a good accuracy in thermal modeling, these methods are very expensive in terms of execution time, especially when the required accuracy of transient temperature is high. Furthermore, accurately modeling the temperature-leakage feedback loop will incur extra invocations of costly thermal simulations. Generally, several days may be needed when a large amount of power configurations need to be examined for evaluating the thermal behavior of software applications or to explore the architectural design space in the early design stage. Such a long simulation time can become prohibitively expensive for computer architects or system designers.

5.1.1 Contributions and chapter organization

To the best of our knowledge, the proposed methodology brings the following novel contributions:

• We develop a learning-based autoregressive (AR) framework to enable fast and accurate transient thermal prediction, specially targeting CMPs. Compared to existing simulation-based models like [7], the proposed framework achieves approximately 113X speed-up, while introducing a root-mean-square-error (RMSE) of only 0.8°C. The proposed framework can be applied to enable a wide spectrum of thermal optimizations or evaluation schemes, such as thermal characterization of software applications and proactive DTM.

• The proposed framework provides concrete, quantitative statistical inferences for the thermal behaviors of a CMP. Somewhat counter-intuitively, the inferences show that the single most important factor to influence the transient temperature is the temperature temporal correlation, rather than its spatial correlation, dynamic power, leakage power or other factors. The temporal correlation can account for approximately 66% of transient temperature changes.

• To demonstrate the effectiveness of our framework, we perform thermal optimization of a CMP by mapping workloads in a thermal-aware fashion. The experimental results show that, compared to the results from a popular thermal-aware mapping similar to [53], the proposed approach can further reduce the peak temperature by 2.9°C on average.

This chapter is organized as follows. Section 5.2 introduces the background thermal knowledge. Section 5.3 provides thermal correlations observed for constructing the proposed AR model. Section 5.4 provides the configurations used in this chapter. Section 5.5 details the proposed AR framework for the transient thermal analyses of CMPs. Section 5.6 demonstrates the experimental results. Section 5.7 provides the discussion and points to the possible future directions.

5.2 Conventional thermal modeling

From a physical perspective, the temperature T is a function of time t and three spatial directions x, y and z. We use $T^t_{x,y,z}$ to denote the temperature T of location (x, y, z) at a certain time point t. $T^t_{x,y,z}$ can be expressed by the heat equation that describes the heat flow in a given homogeneous region over time:

$$\frac{\partial T^t_{x,y,z}}{\partial t} = \vartheta\left(\frac{\partial^2 T^t_{x,y,z}}{\partial x^2} + \frac{\partial^2 T^t_{x,y,z}}{\partial y^2} + \frac{\partial^2 T^t_{x,y,z}}{\partial z^2}\right) + q^t_{x,y,z} \tag{5.1}$$

where ϑ is the material-dependent thermal diffusivity and q is the internally-generated heat [52]. Generally, finite-difference methods (FDM) are used to approximate the partially differentiated terms; for example, the central difference approximation is a popular method to approximate $\partial^2 T/\partial x^2$:

$$\begin{aligned}\frac{\partial^2 T^t_{x,y,z}}{\partial x^2} &= \frac{T^t_{x+1,y,z} - T^t_{x,y,z}}{h^2} - \frac{T^t_{x,y,z} - T^t_{x-1,y,z}}{h^2} + O(h^2) \\ &= \frac{T^t_{x+1,y,z}}{h^2} - \frac{2T^t_{x,y,z}}{h^2} + \frac{T^t_{x-1,y,z}}{h^2} + O(h^2)\end{aligned} \tag{5.2}$$

where h is a sufficiently-small step size used to discretize the continuous variable x. O is the big-O notation [54] used to represent the bound of accuracy loss due to the approximation. To consider the boundary condition, we assume that the environment temperature (or ambient temperature) is set to a given constant value and does not vary over time [7]. $\partial^2 T/\partial t^2$, $\partial^2 T/\partial y^2$ and $\partial^2 T/\partial z^2$ can be derived in a similar manner as Eq(5.2).

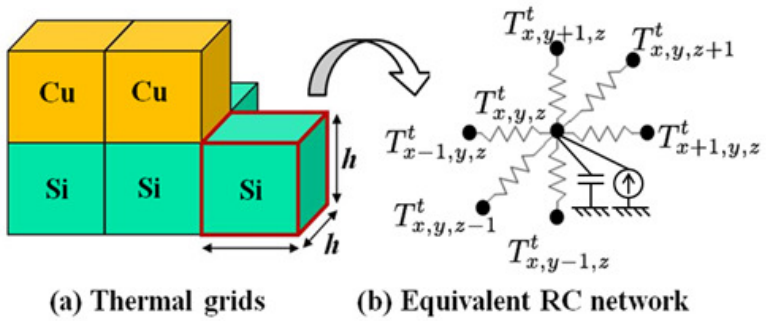


Figure 5.1. Thermal RC model.

There is a well-known analogy between the solid heat conduction and the electrical current flow. The heat conduction can be modeled as a heat current flowing through thermal resistance and capacitance network [52], resulting in temperature differences. The values of thermal RCs depend on the material used to fabricate CMPs. For the purpose of thermal analysis, heat conduction is converted into electrical conduction; CMPs are divided into several cuboidal thermal grids as shown in Figure 5.1(a), and each thermal grid can be converted into an equivalent RC network as shown in Figure 5.1(b), with the temperature modeled as voltage and heat flow modeled as electrical current. Therefore, $T^t_{x,y,z}$ of Eq(5.1) is modeled as the voltage of grid (x, y, z) at the time frame t and q is modeled as the power consumption of a grid. In Figure 5.1(b), we can see that each node connects to six of its immediate neighboring nodes. This is because in many prior arts, such as [7], FDM similar to Eq(5.2), *i.e.*, the central difference, is used to approximate the 2nd-order partial derivatives. The physical meaning behind Eq(5.2) is that "first-level" neighboring grids are used to capture the spatial correlation of temperature changes.

5.3 Thermal correlations

Heat conduction is a continuous process happening within a certain region and over a period of time. This continuous phenomenon determines both spatial and temporal correlations in the temperature differences. More specifically, let us focus on x-y directions and rewrite Eq(5.1) as:

$$\frac{\partial T^t_{x,y}}{\partial t} = \vartheta\left(\frac{\partial^2 T^t_{x,y}}{\partial x^2} + \frac{\partial^2 T^t_{x,y}}{\partial y^2}\right) + q^t_{x,y} \tag{5.3}$$

By using the approximation described in Eq(5.2) on $\partial^2 T/\partial x^2$, $\partial^2 T/\partial y^2$ and $\partial T/\partial t$, we will obtain:

$$\begin{aligned} T^{t+1}_{x,y} = {} & \frac{\vartheta u}{h^2}(T^t_{x\pm1,y} + T^t_{x,y\pm1}) + \frac{h^2 - 4\vartheta u}{h^2}(T^t_{x,y}) + \\ & u(q^t_{x,y}) + O(h^2) \end{aligned} \tag{5.4}$$

where u is the step size for time. Here we assume $h^2 > u$, so the accuracy loss is bounded by $O(h^2)$.

5.3.1 Spatial correlations

The first term of Eq(5.4) represents the spatial correlation of temperature changes, and shows that the first-level neighboring grids are used to approximate $T^{t+1}_{x,y}$. If we further include the second-level neighboring grids, $\partial T/\partial t$ can be expressed as Eq(5.5) by using Taylor's series:

$$T^{t+1}_{x,y} = \sum_{\ell=1}^{2} a_\ell (T^t_{x\pm\ell,y\pm\ell}) + b \cdot (T^t_{x,y}) + c \cdot (q^t_{x,y}) + O(h^4) \tag{5.5}$$

where a_ℓ, b and c are constants derived from ϑ, h and u. Since $h \ll 1$, $O(h^4)$ is smaller than $O(h^2)$ in Eq(5.4), which means the accuracy loss decreases when higher-level neighboring grids are included in the model. Theoretically, when ℓ^{th}-level neighboring grids are included, the accuracy loss should be reduced and bounded by $O(h^{2\ell})$. In practice, a large

ℓ will lead to an extremely-high complexity thermal model. In this thesis, ℓ is empirically set to three to balance the accuracy and the model complexity.

5.3.2 Temporal correlations

The second term of Eq(5.4) or Eq(5.5) shows the temporal correlation between $T_{x,y}^{t+1}$ and $T_{x,y}^{t}$. In Eq(5.4), the step size u needs to be smaller than the thermal RC constant, τ, to guarantee the convergence of the numerical integration. According to [55][7], τ is usually in the range of 0.1 to 0.5ms. In addition, the authors of [7] pointed out that it takes at least 0.1ms to raise the transient temperature of CMPs by 0.1°C. Hence, in this chapter we set the step size u to 0.1ms.

5.4 Configurations and dataset

Before elaborating on the proposed AR framework, we first introduce the architecture and dataset used herein. We introduce the micro-architecture and CMP architecture in Section 5.4.1, followed by the dataset used to train and test the proposed model in Section 5.4.2.

5.4.1 Target architecture

Parameters	**Values**
Number of cores	16
Frequency	3.0 GHz
Technology	45nm node with Vdd =1.0V
L1-I/D caches	64KB, 64B blocks, 2-way SA, LRU
L2 caches	1MB, 64B blocks, 16-way SA, LRU
Pipeline	7 stage deeps, 4 instructions wide

Table 5.1. Processor parameters.

The architecture used throughout this chapter is a symmetric CMP, consisting of 16 out-of-order Alpha 21264 cores [56]. The corresponding micro-architecture parameters are

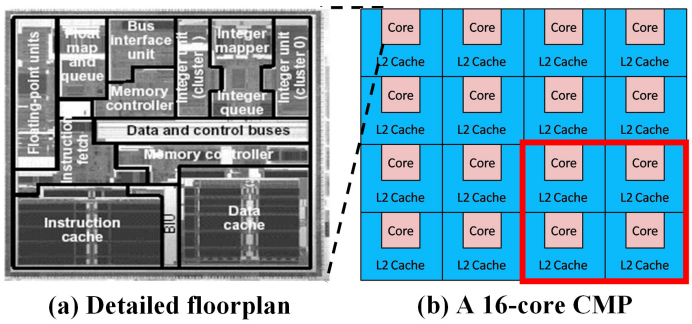


Figure 5.2. An Alpha-core-based CMP and the corresponding floorplan.

listed in Table 5.1. Figure 5.2(a) illustrates the floorplan of Alpha 21264 processing core [56]. This floorplan along with the L2 cache is replicated 16 times in a 4×4 mesh to create a planar 2D CMP. As shown in Figure 5.2(b), processing cores and caches are placed in a fine-grained, interwoven manner. For simplicity and without losing much accuracy, the target CMP is homogeneously partitioned into $32 \times 32 = 1,024$ thermal grids [7]. This resolution (32×32) of thermal grids can be changed according to the different requirements of accuracy. Note that thermal grids are distributed in x-y direction, instead of x-y-z as mentioned in Section 5.2. This is because in the model proposed by Hotspot [7], each grid implicitly includes all vertical components that generate heat, such as metal, active Si and substrates layers.

We use modified SimpleScalar [57], Wattch [6], and Hotspot [7] for the performance, power, and thermal simulations, respectively. We modified the leakage power model in Wattch based on [58][59][60] for more accurate leakage values. Leakage currents are characterized by using HSPICE simulation with the 45nm high performance Predictive Technology Model [61]. For the Hotspot configuration, the chip size and spreader size are set to 0.03m$\times$0.03m; sampling rate is set to 3×10^5 clock cycles; the parameters not mentioned here are assumed to be the default values. SPECcpu2000 benchmarks [62] are randomly selected to form 100 different multi-program workloads for a 16-core CMP. With the above settings, we perform a full-system simulation for 500 million instructions, and then collect the power profiles for the temperature simulation.

5.4.2 Dataset

In this chapter, we use SPECcpu2000 [62] as workloads and the Hotspot [7] as the thermal simulator to characterize the thermal behavior of a CMP. The detailed implementation will be elaborated in Section 5.4.1. The generated thermal responses are used as inputs to train and test the proposed AR model. The dataset contains 100 different power configurations and 513×1024 thermal responses for each power configuration, while 513 is the number of time frames and 1,024 is the number of grids. Each time frame is set to 0.1ms [63]. In this chapter, we treat each grid (x, y) at a time frame t as a sample, so a total of $N = 100 \times 513 \times 1024 \approx 10^7$ samples are used to train and test the proposed AR framework. The features of the dataset is described in Table 5.2. Each sample has P features, including five physical features plus ξ_{AR} autoregressive (AR) features. The five physical features of each sample include: its x location (X), y location (Y), radius (R), total power consumption (P_{tot}), and leakage power consumption (P_{leak}). R is calculated by $\sqrt{(X - mid)^2 + (Y - mid)^2}$ in which mid is set to (32+1)/2 (since the resolution of thermal grids is 32×32). Also, P_{leak} is included in P_{tot}; we separate this term out because P_{leak} is more sensitive to temperature changes [2] and may potentially be a good thermal predictive feature.

Thermal responses	**Samples(Features)**
$100 \times 513 \times 1024$	$100 \times 513 \times 1024 \times (P = 5 + \xi_{AR})$
$conf. \times time \times grids$	$conf. \times time \times grids \times (features)$

Table 5.2. Features of the dataset.

As mentioned in Section 5.3.2, $T^t_{x\pm\ell,y}$, $T^t_{x,y\pm\ell}$ and $T^t_{x,y}$ are highly correlated to $T^{t+1}_{x,y}$, and therefore these features should be included in the dataset to improve the prediction accuracy. These features are called AR features. Unlike the physical features above, AR features will be evaluated at each time frame. Therefore, for each sample, its AR features need to be updated on the fly. ξ_{AR} represents the number of AR features. In this chapter, ξ_{AR} is 13 because $\ell = 3$, such that $T^t_{x\pm\ell,y}$, $T^t_{x,y\pm\ell}$, ℓ = 1 to 3 and $T^t_{x,y}$ are included.

To better explain the proposed methodology, we denote $T^t_{x,y}$ as the thermal response of the i^{th} sample T_i, and both physical and AR features as $\boldsymbol{m}^i = (m_{i1}, \cdots, m_{iP})$. The bold font represents a vector instead of a scalar. Here we focus only on the features of the dataset, which will be used to explain the proposed framework.

5.5 Learning-based AR framework

The learning-based AR framework uses *Lasso* regression [64] to predict T_i. Lasso regression consists of a linear regression with L1 regularization. It shrinks the fitting coefficients and sets some of them to exact zero, and hence tends to retain only the highly relevant features to predict T_i. According to Eq(5.4) and Eq(5.5), T_i can be approximated by a linear function of predictive features $\boldsymbol{m^i}$:

$$T_i = \sum_{j=1}^{P} \alpha_j m_{ij} + \beta \tag{5.6}$$

where $(\boldsymbol{\alpha}, \beta)$ are fitting coefficients. As in the usual regression setup, $\sum_i m_{ij}/N$ are standardized so that $\sum_i m_{ij}/N = 0$ and $\sum_i m_{ij}^2/N = 1$, and T_i are assumed to be conditionally independent given m_{ij} since the potential correlations among T_i are already modeled by AR features in m_{ij}. N is the total number of samples. The physical insight behind Eq(5.6) is that, in addition to the AR features and P_{tot}, the rest of the features are used to linearly converge to $O(h^{2\ell})$.

As we mentioned in Section 2.1, the learning process can be separated into two phases: the training phase and validation phase. The goal of the training phase is to learn the estimate of fitting coefficients $(\boldsymbol{\alpha}, \beta)$, denoted as $(\hat{\boldsymbol{\alpha}}, \hat{\beta})$ and $\hat{\boldsymbol{\alpha}} = (\hat{\alpha_1}, \cdots, \hat{\alpha_P})$. The

$(\hat{\boldsymbol{\alpha}}, \hat{\beta})$ can be learned by:

$$
\begin{aligned}
(\hat{\boldsymbol{\alpha}}, \hat{\beta}) = \quad & \underset{(\boldsymbol{\alpha},\beta)}{arg\,min} \Big\{ \sum_{i=1}^{N} \Big(T_i - \beta - \sum_{j=1}^{P} \alpha_j m_{ij} \Big)^2 \Big\}, \\
& s.t. \sum_{j=1}^{P} |\alpha_j| \leq \lambda.
\end{aligned}
\tag{5.7}
$$

where λ is the parameter to control the amount of shrinkage that is applied to the estimates. In this chapter, the solver provided by [65] is used to optimize Eq(5.7) and learn $(\hat{\boldsymbol{\alpha}}, \hat{\beta})$. Here, we use 10-fold cross validation (CV) to select λ which results in the smallest root-mean-square-error (RMSE): the best value of λ is 1, selected from the range of 10^{-5} to 10^5.

Next we plug learned $(\hat{\boldsymbol{\alpha}}, \hat{\beta})$ into Eq(5.6) and to calculate $\hat{T}_i$ as an estimate of T_i:

$$
\hat{T}_i = \sum_{j=1}^{P} \hat{\alpha}_j m_{ij} + \hat{\beta} \tag{5.8}
$$

By using Eq(5.8), T_i can be calculated instantly if $\boldsymbol{m^i}$ is given. No time-consuming thermal simulation is required. Please note that Eq(5.8) is different from Eq(5.6) because $(\hat{\boldsymbol{\alpha}}, \hat{\beta})$ and $\hat{T}_i$ are estimates, while $(\boldsymbol{\alpha}, \beta)$ and T_i of Eq(5.6) are actual values.

5.5.1 Prediction accuracy

To evaluate the accuracy of the proposed learning model, we again use 10-fold cross validation to calculate the prediction error: Figure 5.3 shows the cross-validated prediction results; the X axis stands for the actual simulated results obtained with Hotspot [7], whereas the Y axis represents the predicted temperatures by using the learning-based AR model. Each color represents one instance of cross validation. As it can be seen in the figure, our thermal prediction is very accurate. The RMSE is 0.43°C and the correlation coefficient (CC) is 0.99. Therefore, just relying on the fitting coefficients $(\hat{\boldsymbol{\alpha}}, \hat{\beta})$ learned

from the proposed framework, one can accurately predict the transient temperature for a CMP, without actually performing time-consuming thermal simulations.

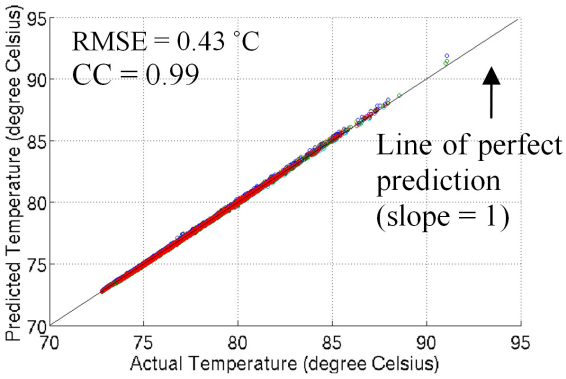


Figure 5.3. Prediction accuracy.

5.5.2 Coefficient analysis

We also show the distribution of fitting coefficients for each feature, namely $\hat{\boldsymbol{\alpha}}$. The physical meaning of $\hat{\boldsymbol{\alpha}}$ is the sensitivity of temperature changes to each predictive feature. Figure 5.4 shows the relative percentage of in a pie chart. All notations here are the same as described in Section 5.5. $T^t_{x\pm\ell,y\pm\ell}$ in Figure 5.4 represents the sum of $\hat{\boldsymbol{\alpha}}$ of $T^t_{x\pm\ell,y\pm\ell}$. We can see that $T^t_{x,y}$ dominates the prediction of $T^{t+1}_{x,y}$ by 66%. This is counter-intuitive because P_{tot} is generally considered as the most important factor to affect temperature. It is also worth mentioning that the $\hat{\boldsymbol{\alpha}}$ of R and P_{leak} are negative values. This is interesting because a grid with a large R actually means that it is located on or close to the rim of a CMP, which has better heat dissipation. Also, a grid with a large P_{leak} means that this grid idles often. Both these two phenomena lead to a lower temperature profile, and hence the corresponding coefficients of R and P_{leak} are negative.

5.6 Forward prediction

So far, we have demonstrated how to predict T_i with $\boldsymbol{m^i}$. Recall that within $\boldsymbol{m^i}$, there are several AR features, such as $T^t_{x,y}$, which cannot be known in advance before the time frame

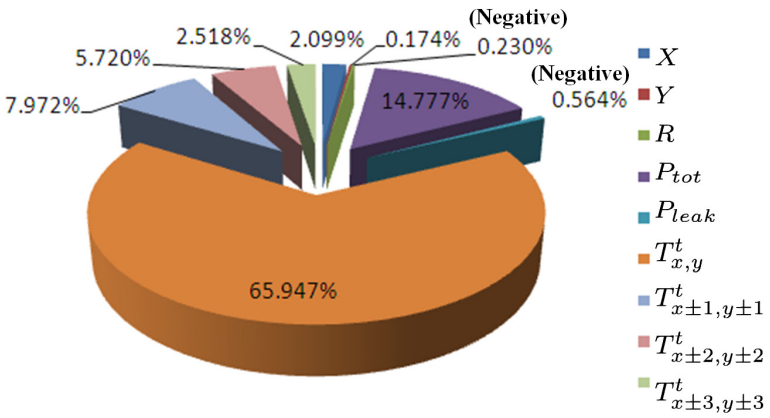


Figure 5.4. Coefficient distribution.

evolves to t. Hence, we need to wait for these AR features to be known, in order to predict $T^{t+1}_{x,y}$. In other words, if we are interested in the transient temperature at the time frame $t+1$, we need to wait until the thermal estimation or measurement, such as the reading from a thermal sensor, at the time frame t is available. This restriction greatly reduces the capability of the proposed AR framework.

To handle the aforementioned problem, we develop a technique called forward prediction. The concept is simple: if $T^t_{x,y}$ is not available yet, but we need it to predict $T^{t+1}_{x,y}$ – we predict $T^t_{x,y}$ first and then use $\hat{T}^t_{x,y}$, namely, the estimate of $T^t_{x,y}$, to predict $T^{t+1}_{x,y}$. The concept can be recursively applied until the time frame equals zero, *i.e.*, all temperature values are the ambient temperature. The computational complexity of this forward prediction is linear with the number of time frame N_t, denoted as $O(N_T)$, and hence can be efficiently computed. With this forward prediction technique, the proposed AR model could be used to predict the transient temperature at any time frame, without be restricted by AR features.

Figure 5.5 shows the RMSE (in Z axis) of each grid (in X axis) over each time frame (in Y axis). For better visualization, we pick 20 of the hottest grids as grids of interest. These grids of interest are often the location of thermal hotspots, so our prediction needs to be accurate there. Note that the RMSE is calculated by using 10-fold cross validation. Generally, the RMSE of each grid stays around 0.7°C, and the highest error in Figure 5.5 is less than 1.1°C. The overall RMSE of every single grid over each time frame is 0.8°C.

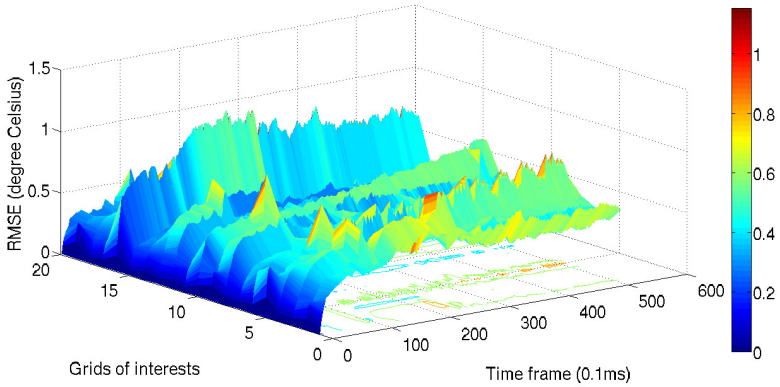


Figure 5.5. Accuracy of forward prediction.

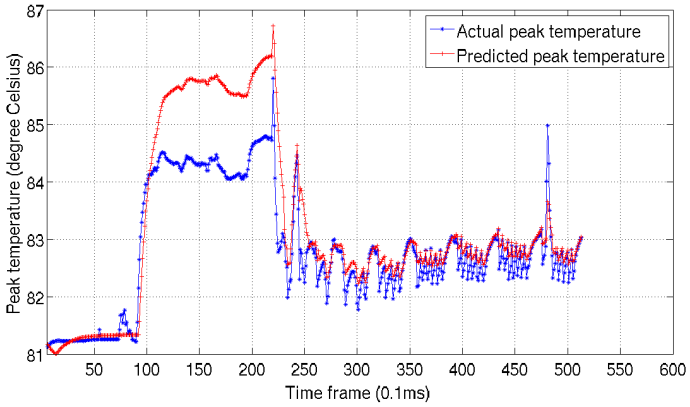


Figure 5.6. Peak temperature prediction.

Also, we are interested in the peak temperature prediction. Figure 5.6 illustrates the peak temperature (in Y axis) of a whole CMP at each time frame t (in X axis) under a "completely clean" power configuration (not involved in the training process of the model). The blue line is the actual temperature obtained via thermal simulation, whereas the red line is the predicted temperature. Although the prediction indeed introduces some errors up to 1.2°C, it is clear that the general trend of peak temperature changes is captured very well by the proposed framework. For execution time, Hotspot [7] needs approximately 291 seconds of CPU time to finish the transient analysis for one power configuration with other settings described in Section 5.4.1. Once the AR model is trained, only 2.57 seconds are needed by

using the forward prediction, and therefore a 113X speed-up is achieved. All these results demonstrate that the proposed forward prediction is accurate and stable.

5.7 Discussion

In this chapter, we present a systematic learning framework that accurately predicts the transient temperature of a CMP by using an AR Lasso model. The proposed model achieves 113X speed-up while introducing a RMSE of only 0.8°C. In [66], the authors applied our proposed model to predict the maximum steady-state temperature of a three-dimensional (3D) CMP, and received a similar prediction error (around 1°C). All these results have demonstrated the high accuracy and robustness of the proposed model. An interesting straight-line future work is to further extend this framework to predict the transient (instead of a steady-state) temperature of a 3D CMP.

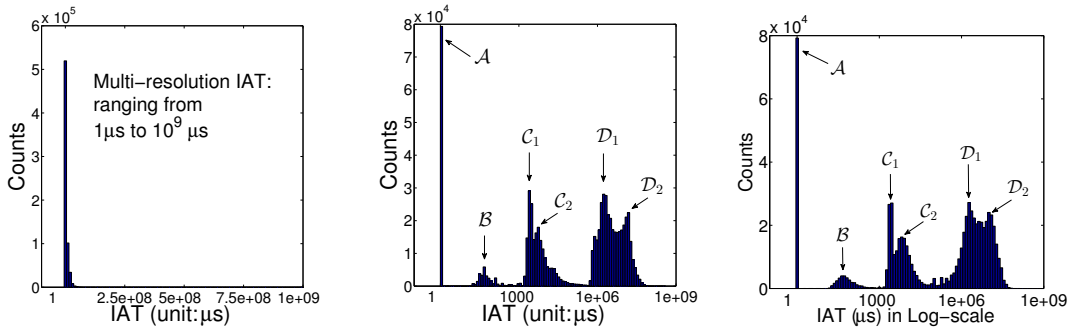
Chapter 6

Modeling job inter-arrivals in a datacenter

A modern datacenter contains tens of thousands of servers, and the computing resources (*e.g.*, processing power and disk storage) are massively-integrated and virtually-shared among these cloud-based services. In this context, the cloud-based scheduling and dynamic resource management directly affect the utilization, energy consumption, and quality of service (QoS) of a datacenter [67]. To design an efficient and effective scheduling algorithm, understanding the patterns of job requests submitted to a datacenter is key. In this chapter, we demonstrate a data-driven approach to characterize and model the distinct patterns of the job submissions.

6.1 Inter-arrival time of job submissions

What are the major characteristics of job inter-arrival process in a datacenter? Could we develop a tool to create synthetic inter-arrivals that match the properties of the empirical data? Understanding the characteristics of job inter-arrivals is the key to design effective scheduling policies to manage massively-integrated and virtually-shared computing resources in a datacenter. Conventionally, during the development of a cloud-based



(a) Empirical IAT in lin. scale (b) Empirical IAT in log scale (c) Synthetic IAT in log scale

Figure 6.1. Deviation from Poisson Process: (a) Histogram of job IAT ($\approx 668,000$ jobs) in linear-scale. (b) Same histogram in log-scale. (c) Synthetic IATs from HIBM. In (a), the histogram has limited number of bins to demonstrate IATs of such a fine-resolution, and the marginal distribution may be misidentified as an (negative) exponential distribution. In (b), *four* distinct clusters can be seen: $\mathcal{A}$: 1μs, $\mathcal{B}$: 10-10^3μs, $\mathcal{C}$: 10^3-10^5μs, and $\mathcal{D}$: 10^6-10^9μs. All four clusters are captured by HIBM as shown in (c).

scheduler, job requests are assumed (1) to be submitted independently and (2) to follow a constant rate λ, which results in a simple and elegant model, Poisson process (PP). PP generates independent and identically distributed (i.i.d.) inter-arrival time (IAT) that follows an (negative) exponential distribution [68]. However, in reality, how much does this inter-arrival process deviate from PP?

To demonstrate how the real inter-arrival process deviates from PP, we use Fig. 6.1 to present the histogram of the IAT for 668,000 jobs submitted and collected in an industrial, large-scale datacenter. The resolution of IAT is 1 microsecond (μs, 10^{-6} sec). As Fig. 6.1(a) shows, the IATs "seem" to follow an (negative) exponential distribution. However, in logarithmic scale as Fig. 6.1(b) shows, surprisingly, *four* distinct clusters (denoted as $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$) with either center-or left-skewed shapes can be seen. This distribution (or a mixture of distributions) clearly does not follow an (negative) exponential distribution, which is always right-skewed in logarithmic scale and therefore cannot create such shapes. This phenomenon has confirmed that the i.i.d. assumption of PP barely holds since certain job requests may depend on one another. For example, a request of disk-backup may immediately be submitted after a request of Gmail service; this dependency violates

the i.i.d assumption and thus invalidates conventional statistical analysis. In this chapter we aim at solving the following two problems:

- **P1: Find patterns.** How to characterize this marginal distribution?

- **P2: Pattern-generating mechanism.** What is a possible mechanism that can generate such job inter-arrivals?

6.1.1 Contributions and chapter organization

In this chapter, we bring the following two contributions:

- **Pattern discovery.** Two key patterns of job inter-arrivals are provided: (1) multiple periodicities and (2) bundling effects. We show the majority (approximately 78%) of job requests show a regular periodicity with a *log-logistic* noise, a skewed, power-law-like distribution. Furthermore, the submission of a job may depend on the occurrence of its previous job, and we refer to this dependency as the *bundling effect*, since these two associated jobs are considered to belong to the same bundle.

- **Generative model.** We propose HIBM, a "HIerarchical Bundling Model," that is succinct and interpretative. HIBM's mathematical expression is succinct that requires only a handful of parameters to create synthetic job inter-arrivals matching the characteristics of empirical data, as shown in Fig. 6.1(c). Furthermore, HIBM has the capability to explain the attribution of the four clusters ($\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$) and the "spikes" ($\mathcal{A}$, $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{D}_1$, and $\mathcal{D}_2$) in Fig. 6.1(b).

The remainder of this chapter is organized as follows. Section 6.2 provides the problem definition, Section 6.3 details the proposed HIBM, Section 6.4 presents the experimental results, and Section 6.5 provides the discussion.

6.2 Problem Definition

In this chapter, we use the trace from Google's cluster [67], which is the first publicly available dataset that presents the diversity and dynamic behaviors of real-world service requests, from a large-scale, multi-purpose datacenter. The trace contains the scheduler requests and actions recorded from 29 days (starting at 19:00 EST, on Sunday May 1st, 2011) of activity in a 12,500-machine cluster. Each request submitted by a user forms a *job* and the trace records approximately 668,000 job submissions.

6.2.1 Terminology and problem formulation

First, we define the terminology used throughout this chapter.

Definition 1 (Job type and job instance)**.** *"Job type" represents a certain type of job that can occur once or multiple times, and "job instance" is the actual occurrence of a job request.*

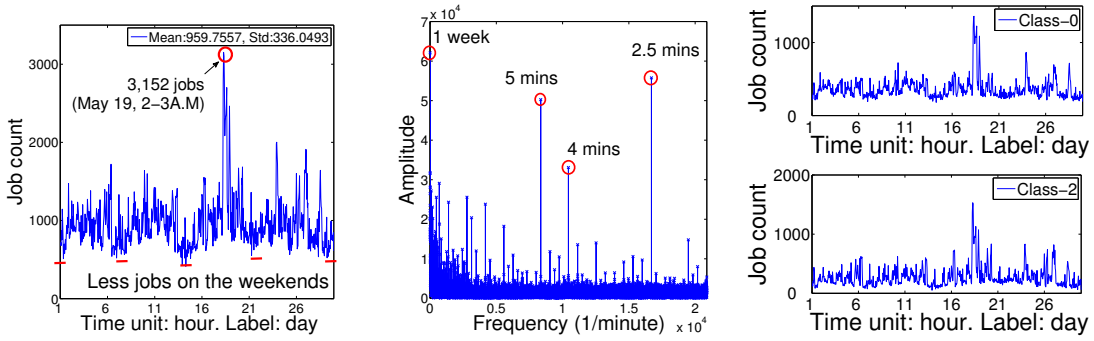
For example, "disk-backup" is a job type that can instantiate several requests; each request (such as "disk-backup at 1:00P.M. on May 2nd") is a job instance.

Definition 2 (Job bundle)**.** *"Job bundle" represents the association of two job types − if two job types are in the same job bundle, the IATs of their job instances will be correlated.*

Like the example used in Section 6.1, two job types "disk-backup" and "Gmail" are functionally-associated, and thus they are considered belonging to the same job bundle. In this case, the inter-arrival of each disk-backup instance will depend on the occurrence of each Gmail instance.

Definition 3 (Job class)**.** *"Job class" represents the priority (or latency sensitiveness) of a job type. In the trace, job class is enumerated as* $\{0, 1, 2, 3\}$ *with a job type of class 3 being the highest priority.*

As mentioned in the Introduction, we have two goals:



(a) Job-instance series (b) Single-sided spectrum (c) Class-0 and Class-2

Figure 6.2. A burst and periodicities: (a) Job instances per hour. A burst (indicated by the red circle) at May 19$^{\text{th}}$ can be observed. (b) Discrete Fourier Transform (DFT) on the job-instance series. The high-amplitude signals correspond to the periods of 1 week, etc. (c) Class-0 (the lowest priority) and class-2 instance series. Notice their similarity (correlation coefficient is 0.94).

- P1: **Find patterns.** Given (1) the job type j, (2) the time stamp of its i^{th} instance (denoted as $t_{j,i}$), and (3) the job class, find the most distinct patterns that are sufficient to characterize the IATs of all job instances in a datacenter.

- P2: **Pattern-generating mechanism.** Given the patterns found in P1, design a model that can generate IATs that match these characteristics of the empirical data and report the model parameters.

6.2.2 Dataset exploration

We begin this section by illustrating the number of job instances over time in Fig. 6.2(a). We collect the time stamp of each job instance when it is first submitted to the datacenter, and then aggregate the total number of job instances within each hour to construct a dataset of one-dimensional time-series. On average, 959.8 job instances are submitted per hour, and in general, less instances are submitted on the weekends whereas more are submitted during weekdays. Interestingly, around 2:00 A.M. on May 19$^{\text{th}}$ (Thursday), a burst of 3,152 job instances can be observed, and its amount is approximately three times higher than the amount on typical Thursday midnights.

Discrete Fourier Transform (DFT) is also performed on the job-instance series. Fig. 6.2(b) provides the amplitude of each discrete frequency, on which we denote four frequencies of high power-spectrum amplitudes: 1-week, 5-min, 4-min and 2.5-min. The reason that the 1-week signal has a high amplitude can be explained by the periodic behavior between weekends and weekdays. Later in Section 6.3.1, we characterize the periodicity and show that both 5-min and 4-min periods can be found during the job inter-arrivals.

6.2.3 Class interdependency

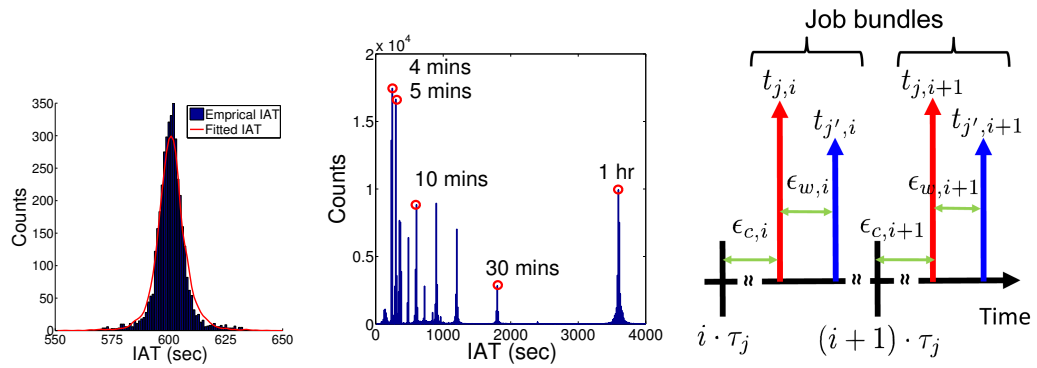
Not all jobs are submitted equal: certain job types have higher priority to be scheduled and executed (class-3, *e.g.*, website services), whereas other jobs do not (class-0, *e.g.* MapReduce workloads) [67].

Observation 1. *The spike $\mathcal{A}$ (1μs) in Fig. 6.1(b) is attributed to the 1μs IAT between a class-0 and a class-2 instance.*

As shown in Fig. 6.2(c), the pattern of class-0 job instances (low priority) is highly similar with the pattern of class-2 instances (high priority), in terms of both trend and quantity. As it can be seen that these instances of class-0 and class-2 contribute to the burst on May 19$^{\text{th}}$ observed in Fig. 6.2(a). Furthermore, the correlation coefficient between class-0 and class-2 instances is 0.94, which makes us think: what is the IAT between a class-0 and a class-2 instance? Surprisingly, this IAT is *exactly 1μs*, which forms the first cluster in Fig. 6.1(b). This phenomenon immediately piques our interest: how to characterize and attribute the rest of three clusters ($\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$) and the corresponding spikes? The answer lies in the "bundling effect" as we will elaborate in Section 6.3.

6.3 HIBM: HIerarchical Bundling Model

In this section, we introduce two major components of HIBM: cross-bundle effects (Section 6.3.1) and within-bundle effects (Section 6.3.2). The complete HIBM framework is presented in Section 6.3.4.



(a) IAT of job type j (b) IAT of all job types (c) Cross-and within-bundle

Figure 6.3. Multiple periodicities: (a) IAT of job type j and fitted PDF by HIBM. (b) IAT of all job types. (c) Illustration of the cross-bundle noise ($\epsilon_{c,i}$) and the within-bundle noise ($\epsilon_{w,i}$) under the period τ_j.

6.3.1 First component: cross-bundle effect

Multiple periodicities To characterize the periodicity of each job type, we first calculate the IAT between every two consecutive job instances of that job type as follows:

$$\delta_{j,i} = t_{j,i} - t_{j,i\text{-}1}, \text{ for } i = 1 \ldots n_j \tag{6.1}$$

where $\delta_{j,i}$ is the i^{th} IAT, $t_{j,i}$ represents the occurrence time of the i^{th} instance of job type j, and n_j is the total number of instances of job type j. Fig. 6.3(a) shows the histogram of such IATs, $\delta_{j,i}$. The histogram is symmetric and has a spike at 600 seconds (10 minutes), which means each instance of job type j arrives approximately every 10 minutes with some noise. Therefore, $t_{j,i}$ can be expressed as:

$$t_{j,i} = i \cdot \tau_j + \epsilon_{c,i} \tag{6.2}$$

where τ_j stands for the period (*e.g.*, 10 minutes in this case) and $\epsilon_{c,i}$ is a random variable representing the "cross-bundle noise." As illustrated in Fig. 6.3(c), the cross-bundle noise ($\epsilon_{c,i}$) represents the delay of a job bundle from its scheduled time ($i \cdot \tau_j$) and in this example

two job types j and j' are in the same bundle. Here, we focus on only the job type j (the red arrows); the within-bundle noise will be elaborated in Section 6.3.2. In this chapter, τ_j is estimated by using the median of IATs of job type j; however, what distribution $\epsilon_{c,i}$ follows remains unclear for now.

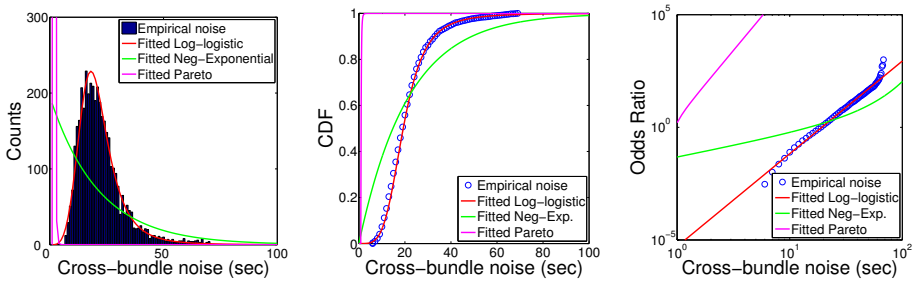
Observation 2. *Multiple periodicities are observed: 4-min, 5-min, 10-min, 15-min, 20-min, 30-min, and 1-hr.*

One question may arise: is this periodic job type a special case, or do IATs of many job types behave like this? To find the answer, we further collect the IATs from all job types and illustrate them by using Fig. 6.3(b). For better visualization, only periods smaller than one hour are demonstrated. In Fig. 6.3(b), multiple periodicities are observed, and the two highest peaks are 4-min and 5-min, which matches the DFT results in Fig. 6.2(b): the frequencies with high amplitudes are 4-min and 5-min. 4-min is also the smallest period that exists in the trace. We would like to point out that the "10-min peak" in Fig. 6.3(b) seems sharper than the peak in Fig. 6.3(a); this is because Fig. 6.3(b) contains several job types that have the same period (10-min), whereas Fig. 6.3(a) contains only one such job type.

Now the question is: what random noise $\epsilon_{c,i}$ will create such IAT distribution shown in Fig. 6.3(a)? Could we use famous "named" distributions, say (negative) exponential or Pareto (power-law), to model this noise?

Modeling cross-bundle noise Among many statistical distributions, we propose to model the cross-bundle noise $\epsilon_{c,i}$ by using Log-logistic distribution (LL), since it is able to model **both the cross-bundle noise and the within-bundle noise** (Section 6.3.2), leading to the unified expression in HIBM. Also, it provides intuitive explanations for sporadic, large delays. The Log-logistic distribution has a power-law tail and its definition is as follows.

Definition 4 (Log-logistic distribution)**.** *Let T be a non-negative continuous random variable and $T \sim LL(\alpha, \beta)$; the cumulative density function (CDF) of a Log-logistic dis-*



(a) Cross-bundle noises (b) CDF (c) Odds Ratio

Figure 6.4. Modeling cross-bundle noise: (a) PDF, (b) CDF (c) Odds Ratio are demonstrated by using Log-logistic, negative-exponential and Pareto distribution, respectively.

tributed variable T *is ,* $CDF(T = t) = F_T(t) = \frac{1}{1+(t/\alpha)^{-\beta}}$*, where* $\alpha > 0$ *is the scale parameter, and* $\beta > 0$ *is the shape parameter. The support* $t \in [0, \infty)$.

Fig. 6.4(a) presents the cross-bundle noise $\epsilon_{c,i}$ and three fitted distributions by using Maximum Likelihood Estimate (MLE) [69]. The distribution shows a left-skewed behavior and sporadically, a few job instances suffer from large delays. This phenomenon is difficult to be captured by distributions with tails decaying exponentially fast (*e.g.*, negative-exponential). On the other hand, the Pareto distribution (a power-law probability distribution), which is also a heavy-tail distribution, lacks the flexibility to model a "hill-shaped" distribution. The goodness-of-fit is tested by using Kolmogorov-Smirnov test [70] with the null hypothesis that the cross-bundle noise is from the fitted Log-logistic distribution. The resulting P-value is 0.2441, and therefore we retain the null hypothesis under the 95% confidence level and conclude that the cross-bundle noise follows Log-logistic distribution.

To better examine the distribution behavior both in the head and tail, we propose to use the Odds Ratio (OR) function.

Lemma 6.3.1.1 (Odds Ratio)**.** *In logarithmic scale, $OR(t)$ has a linear behavior, with a slope β and an intercept ($-\ln\alpha$), if T follows Log-logistic distribution:*

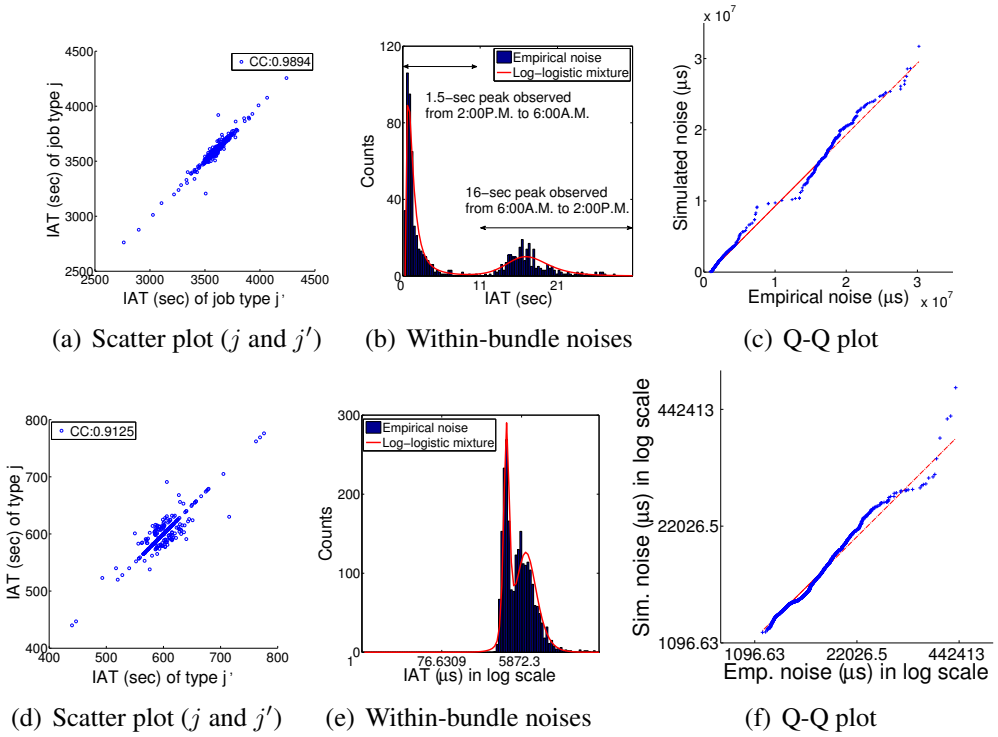
$$OddsRatio(t) = OR(t) = \frac{F_T(t)}{1 - F_T(t)} = \left(\frac{t}{\alpha}\right)^{\beta} \tag{6.3}$$
$$\Rightarrow \ln OR(t) = \beta \ln(t) - \ln\alpha \ \blacksquare$$

As Fig. 6.4(c) shows, the OR of the cross-bundle noise seems to entirely follow the linear line, which serves as another evidence that its marginal distribution follows a Log-logistic distribution. The Log-logistic distribution presents a modified version of the well known phenomenon – "rich gets richer." We conjecture that this phenomenon can be adapted to explain the cross-bundle noise of periodic job instances – "*those delayed long get delayed longer*." If the submission schedule of a job instance is delayed (or preempted) by other jobs with a higher priority, it is likely that this job instance is going to suffer from being further delayed.

6.3.2 Second component: within-bundle effect

Bundling effect and within-bundle noise The bundling effect represents the temporal dependency between two job types j and j'. If the instances of two job types (*e.g.*, Gmail and disk-backup, denoted as job type j and j', respectively) are independent from each other, the correlation coefficient of their IATs should be close to zero. However, as Fig. 6.5(a) shows, IATs of two job types can be highly correlated; the correlation coefficient (CC) is 0.9894. In this context, each $t_{j,i}$ and $t_{j',i}$ must share the same $\epsilon_{c,i}$ due the high correlation. More interestingly, the instances of job type j' always occur after the corresponding instance of j, *i.e.*, $t_{j,i} < t_{j',i}$ as illustrated in Fig. 6.3(c).

We further examine the IAT between job type j and j', namely, $t_{j',i} - t_{j,i}$, referred as "within-bundle noise" ($\epsilon_{w,i}$). The concept of the within-bundle noise also is illustrated by Fig. 6.3(c); furthermore, Fig. 6.5(b) presents a bi-modal distribution of $\epsilon_{w,i}$: one peak



(a) Scatter plot (j and j') (b) Within-bundle noises (c) Q-Q plot

(d) Scatter plot (j and j') (e) Within-bundle noises (f) Q-Q plot

Figure 6.5. HiBM **fits real within-bundle noises: (a) IATs of job type j and j' are highly correlated; the correlation coefficient (CC) is 0.9894. Here, both job type j and j' have the period of 1 hour. (b) Within-bundle noise ($\epsilon_{w,i}$) that creates the spikes $\mathcal{D}_1$ and $\mathcal{D}_2$ can be modeled as a mixture of two Log-logistic distributions. (c) Q-Q plot between the empirical $\epsilon_{w,i}$ and the samples drawn from the fitted Log-logistic mixture. (d)(e)(f) demonstrate another $\epsilon_{w,i}$ in millisecond-scale, and have similar explanations. We would like to point out the spikes $\mathcal{C}_1$ and $\mathcal{C}_2$ can be attributed to the within-bundle noise shown in (e).**

at 1.5-sec observed from 2:00P.M. to 6:00A.M. and the other at 16-sec observed from 6:00A.M. to 2:00P.M.

Observation 3. *The spikes $\mathcal{D}_1$ (1.5sec) and $\mathcal{D}_2$ (16sec) in Fig. 6.1(b) are attributed to HiBM's within-bundle noise in the scale of seconds.*

A possible explanation is that the submissions of job type j' (class 1, latency-insensitive) are delayed or preempted by other high priority job types during the working hours from 6:00A.M. to 2:00P.M., which creates the second mode (the 16-sec peak). Therefore, we model this bi-modal distribution by using a mixture of two Log-logistic distributions. Fig.

6.5(c) shows the Q-Q plot between the empirical $\epsilon_{w,i}$ and samples drawn from the fitted Log-logistic mixture. As it can be seen, each quantile of simulated samples matches the empirical $\epsilon_{w,i}$ very well.

A highly similar situation can be observed from another job bundle, shown in Fig. 6.5(d)(e)(f). Instead of seconds, as Fig. 6.5(e) shows, $\epsilon_{w,i}$ is bi-modal and in the scale of millisecond.

Observation 4. *The spikes $\mathcal{C}_1$ (3ms) and $\mathcal{C}_2$ (5.5ms) in Fig. 6.1(b) are attributed to HiBM's within-bundle noise in the scale of milliseconds.*

In this case, $\epsilon_{w,i}$ can also be modeled by a mixture of two Log-logistic distributions as Fig. 6.5(e)(f) show. For both cases (within-bundle noises in both second-and millisecond-scale), Kolmogorov-Smirnov test is performed; the null hypothesis that $\epsilon_{w,i}$ and the fitted Log-logistic mixture follow the same distribution, is retained under the 95% confidence level. In addition, within-bundle noises are also observed in μs scale, which forms the cluster (and the spike) $\mathcal{B}$ in Fig. 6.1(b) and can also be modeled by the Log-logistic distribution. This is not shown here due to the space limit. Now we are able to explain and model all the clusters and spikes ($\mathcal{B}$, $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{D}_1$ and $\mathcal{D}_2$) with the Log-logistic distribution, leading to the succinctness of HIBM.

Interestingly, even if $\epsilon_{w,i}$ exists, the IATs of job type j and of j' are still highly correlated. The key to create such a phenomenon lies in the hierarchy that cross-bundle noise is always larger than within-bundle noise, $\epsilon_{c,i} > \epsilon_{w,i}$. In the trace, the scale of $\epsilon_{c,i}$ is approximately in the magnitude of minutes, whereas $\epsilon_{w,i}$ is in the magnitude of seconds, milliseconds or even microseconds. Based on this observation, we propose a unified model to describe the IATs of two job types in the same bundle, which serves as the backbone of the proposed HIBM:

$$\begin{cases} t_{j,i} = i \cdot \tau_j + \epsilon_{c,i} \\ t_{j',i} = t_{j,i} + \epsilon_{w,i} = i \cdot \tau_j + \epsilon_{c,i} + \epsilon_{w,i} \end{cases} \tag{6.4}$$

where $\epsilon_{c,i} \sim LL(\alpha_{c,\kappa}, \beta_{c,\kappa})$, $\epsilon_{w,i} \sim$ a mixture of two LL distributions, expressed as:

$$\epsilon_{w,i} \sim p_{w,\kappa} \cdot LL(\alpha_{w,\kappa}, \beta_{w,\kappa}) + (1 - p_{w,\kappa}) \cdot LL(\alpha_{w',\kappa}, \beta_{w',\kappa}) \tag{6.5}$$

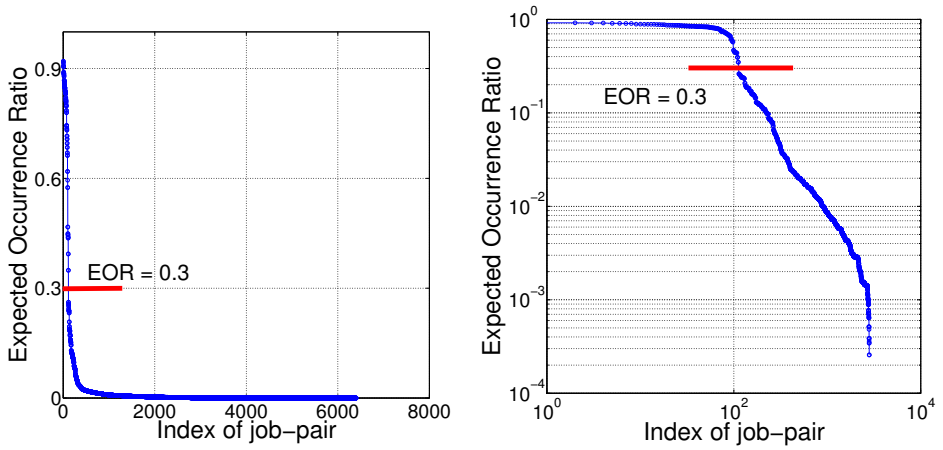
$p_{w,\kappa} \in [0, 1]$, $\kappa \in \{\mathcal{B}, \mathcal{C}, \mathcal{D}\}$. Given the empirical data, $\alpha_{c,\kappa}$, $\beta_{c,\kappa}$ can be estimated by MLE and $p_{w,\kappa}$, $\alpha_{w,\kappa}$, $\beta_{w,\kappa}$, $\alpha_{w',\kappa}$, $\beta_{w',\kappa}$ can be estimated by Expectation Maximization (EM) [69].

Bundle detection algorithm After explaining the bundling effect, the next question is how to determine if two certain job types belong to the same job bundle. We ask: given each pair of $t_{j,i}$ and $t_{j',i}$, how do we know these IATs, namely, $|t_{j,i} - t_{j',i}|$, are caused by within-bundle noises ($\epsilon_{w,i}$), or just coincidentally by a job instance occurring closely to another instance? What if two job types have different periods? To answer these questions, we propose a metric "expected occurrence ratio" (EOR) that compares the empirical counts and the expected counts of within-bundle noises. EOR $\in [0, 1]$ and a high EOR value indicates that job type j and j' are likely to be in the same job bundle. The details of the proposed EOR are in Section 6.3.3. The intuition is similar to hypothesis testing. We examine the EOR between each pair of job types and illustrate it by using Fig 6.6; the majority of pairs have EOR less than 0.3, whereas other few pairs have EOR very close to 0.8. In this chapter, we select an EOR of 0.3 as threshold and therefore two job types are considered unbundled if their EOR is less than 0.3.

6.3.3 Expected occurrence ratio

Here, we elaborate on using the expected occurrence ratio (EOR) to determine if two job types are bundled. The expected occurrence ratio (EOR) of job type j and j' can be calculated as:

$$EOR(j, j') = \mathcal{N}_\kappa \cdot \left(\frac{\mathcal{T}}{LCM(\tau_j, \tau_{j'})} \cdot \rho_j \cdot \rho_{j'} \right)^{-1} \tag{6.6}$$



(a) EOR in linear scale (b) EOR in logarithmic scale

Figure 6.6. Expected occurrence ratio (EOR): the EOR between each pair of job types is examined. The majority of pairs have EOR less than 0.3, whereas other few pairs have EOR very close to 0.8.

where $\mathcal{N}_\kappa$ represents the number of the IATs occurred in the range of the cluster $\kappa \in \{\mathcal{B}, \mathcal{C}, \mathcal{D}\}$ in Fig. 6.1(b), $\mathcal{T}$ is the total duration, $LCM(\tau_j, \tau_{j'})$ is the Least Common Multiple (LCM) between two periods τ_j and $\tau_{j'}$, finally ρ_j and $\rho_{j'}$ are the missing rates of job type j and j', respectively. The intuition of EOR is similar to hypothesis testing: to compare the empirical count of IATs in the range of a certain cluster with its expected count. The expected count, $\frac{\mathcal{T}}{LCM(\tau_j, \tau_{j'})} \cdot \rho_j \cdot \rho_{j'}$, is calculated under the assumption that job type j and j' are bundled. Therefore, if j and j' are actually bundled, the value of $\mathcal{N}_\kappa$ will be very close to the expected count, resulting in an EOR ≈ 1. On the other hand, if EOR is very close to 0, j and j' are considered unbundled, since the observed $\mathcal{N}_\kappa$ just occurred by coincidence.

6.3.4 Complete HIBM framework

By assembling the cross-bundle effect (Section 6.3.1) and the within-bundle effect (Section 6.3.2) together, we describe here the complete HIBM framework by using Algorithm 1. The inputs to HIBM are user-defined periods, the total duration $\mathcal{T}$, and the parameters of

Algorithm 1: HIBM Generation

Result: Inter-arrival process of job instances, $t_{j,i}$ for all j and i, given periods τ_j for each job type j, total duration $\mathcal{T}$, $\alpha_{c,\kappa}$, $\beta_{c,\kappa}$, $p_{w,\kappa}$, $\alpha_{w,\kappa}$, $\beta_{w,\kappa}$, $\alpha_{w',\kappa}$, and $\beta_{w',\kappa}$.

initialization: $JS = []$;

for *each* j **do**

  for i = *1 to* $\left\lfloor \frac{\mathcal{T}}{\tau_j} \right\rfloor$ **do**

    if *job type* j *is bundled with job type* j' **then**

      $t_{j,i} = t_{j',i} + \epsilon_{w,i}$,

      $\epsilon_{w,i} \sim p_{w,\kappa} \cdot LL(\alpha_{w,\kappa}, \beta_{w,\kappa}) + (1 - p_{w,\kappa}) \cdot LL(\alpha_{w',\kappa}, \beta_{w',\kappa})$;

    else

      $t_{j,i} = i \cdot \tau_j + \epsilon_{c,i}, \epsilon_{c,i} \sim LL(\alpha_{c,\kappa}, \beta_{c,\kappa})$;

    end

    JS = JS appending $t_{j,i}$;

  end

end

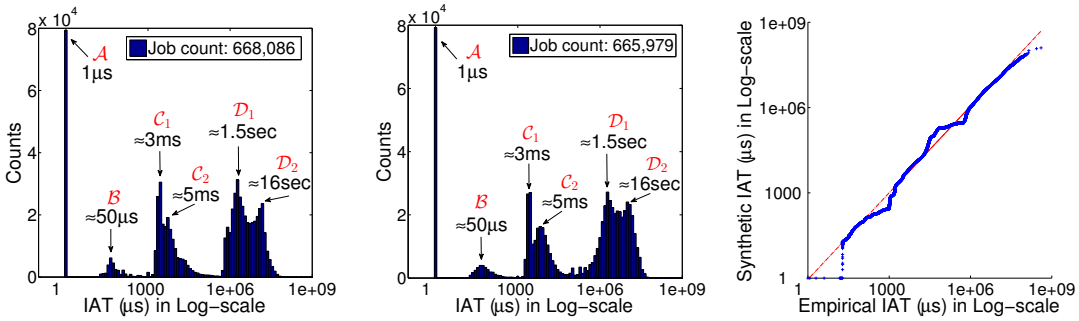
Sort JS in ascending order;

return JS;

Log-logistic distributions as described in Eq (6.4). In our case, the periods are set according to the empirical data as shown in Fig. 6.3(b), the $\mathcal{T}$ is set to one month as mentioned in Section 6.2.2, and the parameters described in Eq (6.4) are estimated by MLE and EM. For each job type j, HIBM calculates its total number of instances by $\left\lfloor \frac{\mathcal{T}}{\tau_j} \right\rfloor$. Next, for the i^{th} instance of job type j, there will be two possible cases: (1) $t_{j,i}$ is bundled with $t_{j',i}$ or (2) $t_{j,i}$ is in its own job bundle (not bundled with any other job type). In the first case, $t_{j,i}$ is estimated according to Eq (6.2), whereas in the second case, $t_{j,i}$ is estimated according to Eq (6.4). The estimated $t_{j,i}$ is recorded in JS for all j and i. Finally, JS is sorted in ascending order and then HIBM outputs JS as job inter-arrivals.

6.4 Experimental Results

We validate HIBM by using the empirical data. The comparisons between the synthetic IATs generated by HIBM and empirical IATs are illustrated by Fig. 6.7. Fig. 6.7(a)(b) present the histogram of the empirical IATs and the synthetic IATs side by side. As it can be seen, the synthetic IATs match the distinct characteristics of the empirical IATs: the



(a) Empirical IATs (b) Synthetic IATs (c) Q-Q plot

Figure 6.7. Comparisons between Synthetic IATs and the empirical IATs: (a) Histogram of empirical IATs in log scale. (b) Histogram of synthetic IATs in log scale. (c) Q-Q plot. The synthetic IATs generated by HIBM match the characteristics of the empirical IATs: the job-instance counts (only 0.3% difference), the four clusters, and all the spikes ($\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}_1$, $\mathcal{C}_2$ $\mathcal{D}_1$, and $\mathcal{D}_2$). In addition, each quantile of the synthetic IATs matches the corresponding quantile from the empirical data very well.

job-instance counts (only 0.3% difference), the four clusters, and all the spikes ($\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}_1$, $\mathcal{C}_2$ $\mathcal{D}_1$, and $\mathcal{D}_2$). Fig. 6.7(c) presents the Q-Q plot, from which we can also observe that each quantile of the synthetic IATs matches the corresponding quantile from the empirical data very well.

We begin the discussion with HIBM's succinctness. HIBM requires only a handful of parameters as described in Algorithm 1 to generate job inter-arrivals that match the characteristics from the empirical data, even when the i.i.d. assumption is violated – the submissions of certain instances depend on one another. Therefore, HIBM can be used as a tool to create more realistic job inter-arrivals to design, evaluate, and optimize the cloud-based scheduler of a datacenter.

Also thanks to HIBM's interpretability, we now understand the four distinct clusters observed from the empirical data can be attributed to both class interdependency ($\mathcal{A}$: 1μs) and within-bundle noises ($\mathcal{B}$: 10-10^3μs, $\mathcal{C}$:10^3-10^5μs, and $\mathcal{D}$:10^6-10^9μs). In addition, the 3ms and 5ms spikes ($\mathcal{C}_1$ and $\mathcal{C}_2$) can be attributed to the within-bundle noise shown in Fig. 6.5(e), and similarly 1.5sec and 16sec spikes ($\mathcal{D}_1$ and $\mathcal{D}_2$) can be attributed to the within-bundle noise shown in Fig. 6.5(b). Furthermore, the cross-bundle noises in HIBM provides

intuitive explanation − "those delayed long get delayed longer" − for the delays occurred on periodic job instances.

6.5 Discussion

In this chapter, we investigate and analyze the inter-arrivals of job requests in an industrial, large-scale datacenter. Two main contributions are summarized below:

- **Pattern discovery.** We discover two key patterns of job inter-arrivals: (a) multiple periodicities and (b) bundling effects. In addition, we propose to use Log-logistic distributions to model both cross-bundle and within-bundle noises.

- **Generative model.** We propose HIBM, a succinct and interpretative model. HIBM requires only a handful of parameters to generate job inter-arrivals mimicking the empirical data. In addition, HIBM also attributes the four distinct clusters and the corresponding spikes to both within-bundle noises and class interdependency, and provides intuitive explanation "those delayed long get delayed longer" to the cross-bundle noises of periodic job types.

We further point out that the periodicity and bundling effects described in this chapter violate the i.i.d. assumption used by Poisson process. Therefore, instead of the Poisson process, the proposed HIBM should be used to evaluate the effectiveness of a cloud-based scheduler. One interesting and useful extension of HIBM is to associate job inter-arrivals with the usage of computing resources (such as CPU time and disk I/O). Such model provides a more comprehensive view of job properties, which may facilitate the development of scheduling algorithm for further improving QoS or energy efficiency.

Chapter 7

Related work

Learning-based or data-driven methods are not the only set of approaches for modeling performance or energy efficiency. In this chapter, we discuss about the prior arts that have addressed the related issues from more diverse viewpoints. We also provide the literature that applies learning-based frameworks for modeling process variations and other metric (or design) of interests.

7.1 Performance and power modeling

There is a large body of work that has been proposed to apply conventional DVFS to reduce power consumption for multi-core processors. Herbert *et al.* [71] have proposed and evaluated DVFS policies for chip multi-processors (CMPs) to address both workload and process variations. Li *et al.* [72] used both DVFS and a variable number of cores to achieve the most power efficient operating point. Going beyond DVFS for processing cores, Mishra *et al.* [73] proposed a DVFS strategy specifically targeting the communication fabric of CMPs. From an implementation perspective, Park *et al.* [74] introduced a framework to accurately estimate the overhead of DVFS including voltage regulator and inductive losses. Mazumdar *et al.*[75] proposed a novel voltage-stacking technique to reduce the supply current of CMPs.

Recently, reinforcement learning (RL) and machine learning (ML) have emerged as popular and robust power management schemes due to their adaptive properties [76, 77, 78, 79, 80]. However, none of the above work has addressed and evaluated the effectiveness of using RL on the power management for NoC-based CMPs, while also including uncore resources. The power management of uncore only or multi-processor systems-on-chip (MPSoC) has also drawn lots of attention from both industry and academia [81, 51, 82, 83]. Although the power management for cores, uncores or MPSoCs has been extensively studied and discussed, none of the previous work has considered synergistic DVFS for NoC-based CMPs to maximize the performance under iso-power conditions.

7.2 Wide-operating range: from NTC to TB

In the context of NTC, Torrellas [84] has evaluated using near-threshold operations on extreme-scale computing. Kaul *et al.* [15] and Jain *et al.* [19] demonstrated novel processor designs that can be operated under a wide range of voltage and frequency. Jain *et al.* [19] further showed that the best energy efficiency (without any constraint) can be obtained in the near-threshold region. Dreslinski *et al.* [16] provided a complete analysis of NTC on multi-core processors.

Regarding the TB mode of operation, Isci *et al.* [18] analyzed the benefit of maximizing performance under a predefined power budget. Cochran *et al.* [85] developed an algorithm to maximize per-thread performance. Very recently, Juan *et al.* [10] adapted reinforcement learning to perform adaptive RDVFS on CMPs. However, none of the above works have addressed the optimization framework for DVFS over a wide-operating range that includes both NTC and TB.

7.3 Thermal dynamics

Thermal modeling for CMPs has received a lot of attention recently. Huang *et al.* [7] proposed Hotspot−an accurate, simulation-based thermal model for planar ICs−and the

corresponding thermal-aware floorplanning. Li *et al.* [86] developed an efficient numerical method to solve large thermal grids for ICs. Bosch [87] demonstrated a thermal model with special focus on the heat flux distribution over the sides of a component. Wang *et al.* [88] proposed a transient thermal simulator based on an alternating direction implicit method. All these work share a common point: the thermal models constructed are physics-based instead of data-driven.

7.4 Learning process variations and Network-on-Chip (NoC)

In addition to performance, power, and thermal dynamics of a multi-core systems, learning-based (or data-driven) concepts have also been applied to model other aspect of interests. Juan *et al.* [9, 89] have demonstrated a statistical framework to model the effects of manufacturing process variations, with a special focus on leakage and thermal variability. The corresponding mitigation techniques have also been presented in [9, 89]. Qian *et al.* [90] has proposed SVR-NoC, using support vector regression (SVR) for evaluating Network-on-Chip (NoC) latency performance.

Chapter 8

Conclusion

In this thesis, we have demonstrated that the performance, power and thermal characteristics of a multi-core system can be well-modeled by Gray-box computing: the concept of data-driven approaches incorporating domain knowledge. Mathematically, the domain knowledge is included by using (1) Gray-box model or (2) Bayesian inference. This chapter concludes by summarizing the key results of this thesis and presenting avenues for further research.

8.1 Learning the performance and power of a multi-core system

For a multi-core system, the performance and power relationship can be learned by a constrained-posynomial function, with the average accuracy 96%. The learned function provides convexity for further multi-constrained optimization: (1) an additional 13.28% of the energy consumption is reduced under iso-performance conditions, and (2) the throughput is increased by additional 7.54% under iso-power conditions.

Regarding to the multi-core system equipped with network-on-chip as the communication backbone, reinforcement learning (RL) is an effective mechanism for synergic DVFS of cores and uncore. The parameters used in the Markov Decision Process for the RL are

learned via Dirichlet Prior and Posterior. The experimental results show that the average performance is increased by 10.9%.

8.2 Learning the thermal dynamics of a multi-core system

The thermal dynamics of a multi-core system are directly reflected on the operating temperature. We have proposed to model the operating temperature via Lasso regression−an linear regression with L1 norm regularizer. The proposed model achieves more than 98% accuracy and serves as a fast alternative (113X speedup) of the conventional thermal simulation.

8.3 Learning the job inter-arrivals of a datacenter

Characterizing the inter-arrival pattern of jobs submitted to a datacenter is the key to design a cloud-based scheduling algorithm for improving the utilization and quality of services (QoS). The proposed HIBM is a succinct and interpretative model. HIBM requires only a handful of parameters to generate job inter-arrivals mimicking the empirical data. In addition, HIBM also attributes the four distinct clusters and the corresponding spikes to both within-bundle noises and class interdependency, and provides intuitive explanation "those delayed long get delayed longer" to the cross-bundle noises of periodic job types.

8.4 Future work: learning the usage dynamics of a datacenter

One straight-line future work is to adapt Gray-Box computing to learn a function that associates a job submission with the computing resource consumed (CPU time, memory access, disk I/O). Such a function can be used in synergy with HIBM to provide system architects a better view of how to design an efficient and effective scheduling algorithm for a datacenter.

Bibliography

[1] Robert H Dennard, Fritz H Gaensslen, V Leo Rideout, Ernest Bassous, and Andre R LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *Solid-State Circuits, IEEE Journal of*, 9(5):256–268, 1974.

[2] David Brooks, Robert P Dick, Russ Joseph, and Li Shang. Power, thermal, and reliability modeling in nanometer-scale microprocessors. *Micro, IEEE*, 27(3):49–62, 2007.

[3] Da-Cheng Juan, Yu-Ting Chen, Ming-Chao Lee, and Shih-Chieh Chang. An efficient wake-up strategy considering spurious glitches phenomenon for power gating designs. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 18(2):246–255, 2010.

[4] Albert Greenberg, James Hamilton, David A Maltz, and Parveen Patel. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM Computer Communication Review*, 39(1):68–73, 2008.

[5] Milo MK Martin, Daniel J Sorin, Bradford M Beckmann, Michael R Marty, Min Xu, Alaa R Alameldeen, Kevin E Moore, Mark D Hill, and David A Wood. Multifacet's general execution-driven multiprocessor simulator (gems) toolset. *ACM SIGARCH Computer Architecture News*, 33(4):92–99, 2005.

[6] David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *ACM SIGARCH Computer Architecture News*, volume 28, pages 83–94. ACM, 2000.

[7] Wei Huang, Shougata Ghosh, Sivakumar Velusamy, Karthik Sankaranarayanan, Kevin Skadron, and Mircea R Stan. Hotspot: A compact thermal modeling methodology for early-stage vlsi design. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(5):501–513, 2006.

[8] Sheng Li, Jung Ho Ahn, Richard D Strong, Jay B Brockman, Dean M Tullsen, and Norman P Jouppi. Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 469–480. IEEE, 2009.

[9] Da-Cheng Juan, Siddharth Garg, and Diana Marculescu. Statistical thermal evaluation and mitigation techniques for 3d chip-multiprocessors in the presence of process

variations. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pages 1–6. IEEE, 2011.

[10] Da-Cheng Juan and Diana Marculescu. Power-aware performance increase via core/uncore reinforcement control for chip-multiprocessors. In *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, pages 97–102. ACM, 2012.

[11] Da-Cheng Juan, Huapeng Zhou, Diana Marculescu, and Xin Li. A learning-based autoregressive model for fast transient thermal analysis of chip-multiprocessors. In *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, pages 597–602. IEEE, 2012.

[12] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[13] Larry Wasserman. *All of statistics: a concise course in statistical inference*. Springer, 2003.

[14] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.

[15] Himanshu Kaul, Mark A Anders, Sanu K Mathew, Steven K Hsu, Amit Agarwal, Ram K Krishnamurthy, and Shekhar Borkar. A 320 mv 56 μw 411 gops/watt ultra-low voltage motion estimation accelerator in 65 nm cmos. *Solid-State Circuits, IEEE Journal of*, 44(1):107–114, 2009.

[16] Ronald G Dreslinski, Michael Wieckowski, David Blaauw, Dennis Sylvester, and Trevor Mudge. Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits. *Proceedings of the IEEE*, 98(2):253–266, 2010.

[17] James Charles, Preet Jassi, Narayan S Ananth, Abbas Sadat, and Alexandra Fedorova. Evaluation of the intel® core? i7 turbo boost feature. In *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*, pages 188–197. IEEE, 2009.

[18] Canturk Isci, Alper Buyuktosunoglu, Chen-Yong Cher, Pradip Bose, and Margaret Martonosi. An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. In *Proceedings of the 39th annual IEEE/ACM international symposium on microarchitecture*, pages 347–358. IEEE Computer Society, 2006.

[19] S. Jain *et al.* A 280mv-to-1.2v wide-operating-range ia-32 processor in 32nm cmos. *ISSCC*, 2012.

[20] C.M. Bishop. Pattern recognition and machine learning. *Springer*, 2006.

[21] G. Casella *et al.* Statistical inference. *Cengage learning*, 2001.

[22] C. Bienia *et al.* The parsec benchmark suite: Characterization and architectural implications. *PACT*, 2008.

[23] S. Woo *et al.* The splash-2 programs: Characterization and methodological considerations. 1995.

[24] T.E. Carlson *et al.* Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. *SC*, 2011.

[25] C. Enz *et al.* An analytical mos transistor model valid in all regions of operation and dedicated to low-voltage and low-current applications. *Analog Integrated Circuits and Signal Processing*, 1995.

[26] D. Markovic *et al.* Ultralow-power design in near-threshold region. *Proceedings of the IEEE*, 2010.

[27] Y. Cao *et al.* Predictive technology model. *http://ptm.asu.edu/*, 2012.

[28] S. Herbert *et al.* Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. *ISLPED*, 2007.

[29] K. Bowman *et al.* Impact of die-to-die and within-die parameter variations on the throughput distribution of multi-core processors. *ISLPED*, 2007.

[30] L. Wasserman. All of statistics: A concise course in statistical inference. *Springer*, 2003.

[31] S. Boyd *et al.* Convex optimization. *Cambridge university press*, 2004.

[32] V. Nookala *et al.* Temperature-aware floorplanning of microarchitecture blocks with ipc-power dependence modeling and transient analysis. *ISLPED*, 2006.

[33] W Bircher, Jason Law, Madhavi Valluri, and Lizy K John. Effective use of performance monitoring counters for run-time prediction of power. *University of Texas at Austin Technical Report TR-041104-01*, 2004.

[34] S. Li *et al.* The mcpat framework for multicore and manycore architectures: Simultaneously modeling power, area, and timing. *TACO*, 2013.

[35] T. Miller *et al.* Booster: Reactive core acceleration for mitigating the effects of process variation and application imbalance in low-voltage chips. *HPCA*, 2012.

[36] E. Le Sueur *et al.* Dynamic voltage and frequency scaling: The laws of diminishing returns. *ICPACS*, 2010.

[37] Radu Marculescu, Umit Y Ogras, Li-Shiuan Peh, Natalie Enright Jerger, and Yatin Hoskote. Outstanding research problems in noc design: system, microarchitecture, and circuit perspectives. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(1):3–21, 2009.

[38] Luca Benini and Giovanni De Micheli. Networks on chips: A new soc paradigm. *Computer*, 35(1):70–78, 2002.

[39] Alaa R Alameldeen and David A Wood. Ipc considered harmful for multiprocessor workloads. *Micro, IEEE*, 26(4):8–17, 2006.

[40] Reetuparna Das, Asit K Mishra, Chrysostomos Nicopoulos, Dongkook Park, Vijaykrishnan Narayanan, Ravishankar Iyer, Mazin S Yousif, and Chita R Das. Performance and power optimization through data compression in network-on-chip architectures. In *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, pages 215–225. IEEE, 2008.

[41] Niket Agarwal, Tushar Krishna, Li-Shiuan Peh, and Niraj K Jha. Garnet: A detailed on-chip network model inside a full-system simulator. In *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, pages 33–42. IEEE, 2009.

[42] James Charles, Preet Jassi, Narayan S Ananth, Abbas Sadat, and Alexandra Fedorova. Evaluation of the intel® core i7 turbo boost feature. In *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*, pages 188–197. IEEE, 2009.

[43] Paul Sweazey and Alan Jay Smith. A class of compatible cache consistency protocols and their support by the ieee futurebus. In *ACM SIGARCH Computer Architecture News*, volume 14, pages 414–423. IEEE Computer Society Press, 1986.

[44] Peter S Magnusson, Magnus Christensson, Jesper Eskilson, Daniel Forsgren, Gustav Hallberg, Johan Hogberg, Fredrik Larsson, Andreas Moestedt, and Bengt Werner. Simics: A full system simulation platform. *Computer*, 35(2):50–58, 2002.

[45] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. The parsec benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 72–81. ACM, 2008.

[46] Andrea Bartolini, Matteo Cacciari, Andrea Tilli, Luca Benini, and Matthias Gries. A virtual platform environment for exploring power, thermal and reliability management control strategies in high-performance multicores. In *Proceedings of the 20th symposium on Great lakes symposium on VLSI*, pages 311–316. ACM, 2010.

[47] Hang-Sheng Wang, Xinping Zhu, Li-Shiuan Peh, and Sharad Malik. Orion: a power-performance simulator for interconnection networks. In *Microarchitecture, 2002.(MICRO-35). Proceedings. 35th Annual IEEE/ACM International Symposium on*, pages 294–305. IEEE, 2002.

[48] Andrew B Kahng, Bin Li, Li-Shiuan Peh, and Kambiz Samadi. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In *Proceedings of the conference on Design, Automation and Test in Europe*, pages 423–428. European Design and Automation Association, 2009.

[49] Shyamkumar Thoziyoor, Jung Ho Ahn, Matteo Monchiero, Jay B Brockman, and Norman P Jouppi. A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies. In *Computer Architecture, 2008. ISCA'08. 35th International Symposium on*, pages 51–62. IEEE, 2008.

[50] Sebastian Herbert and Diana Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*, pages 38–43. IEEE, 2007.

[51] Asit K Mishra, Reetuparna Das, Soumya Eachempati, Ravi Iyer, Narayanan Vijaykrishnan, and Chita R Das. A case for dynamic frequency tuning in on-chip networks. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 292–303. IEEE, 2009.

[52] Michael Reed and Barry Simon. Methods of modern mathematical physics. *Functional analysis*, 1, 1972.

[53] Ayse K Coskun, Richard Strong, Dean M Tullsen, and Tajana Simunic Rosing. Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, pages 169–180. ACM, 2009.

[54] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2001.

[55] Kevin Skadron, Tarek Abdelzaher, and Mircea R Stan. Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management. In *High-Performance Computer Architecture, 2002. Proceedings. Eighth International Symposium on*, pages 17–28. IEEE, 2002.

[56] Richard E Kessler. The alpha 21264 microprocessor. *Micro, IEEE*, 19(2):24–36, 1999.

[57] Doug Burger and Todd M Austin. The simplescalar tool set, version 2.0. *ACM SIGARCH Computer Architecture News*, 25(3):13–25, 1997.

[58] J Adam Butts and Gurindar S Sohi. A static power model for architects. In *Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture*, pages 191–201. ACM, 2000.

[59] Lerong Cheng, Puneet Gupta, Costas J Spanos, Kun Qian, and Lei He. Physically justifiable die-level modeling of spatial variation in view of systematic across wafer variability. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 30(3):388–401, 2011.

[60] Stefan Rusu, Simon Tam, Harry Muljono, David Ayers, Jonathan Chang, Brian Cherkauer, Jason Stinson, John Benoit, Raj Varada, Justin Leung, et al. A 65-nm dual-core multithreaded xeon® processor with 16-mb l3 cache. *Solid-State Circuits, IEEE Journal of*, 42(1):17–25, 2007.

[61] Wei Zhao and Yu Cao. Predictive technology model for nano-cmos design exploration. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 3(1):1, 2007.

[62] John L Henning. Spec cpu2000: Measuring cpu performance in the new millennium. *Computer*, 33(7):28–35, 2000.

[63] Kevin Skadron, Mircea R Stan, Karthik Sankaranarayanan, Wei Huang, Sivakumar Velusamy, and David Tarjan. Temperature-aware microarchitecture: Modeling and implementation. *ACM Transactions on Architecture and Code Optimization (TACO)*, 1(1):94–125, 2004.

[64] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[65] Mark Schmidt. Least squares optimization with l1-norm regularization. *Project Report, University of British Columbia*, 2005.

[66] Da-Cheng Juan, Siddharth Garg, and Diana Marculescu. Statistical peak temperature prediction and thermal yield improvement for 3d chip-multiprocessors. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 2014.

[67] Charles Reiss, Alexey Tumanov, Gregory R Ganger, Randy H Katz, and Michael A Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the Third ACM Symposium on Cloud Computing*, page 7. ACM, 2012.

[68] Wolfgang Fischer and Kathleen Meier-Hellstern. The markov-modulated poisson process (mmpp) cookbook. *Performance Evaluation*, 18(2):149–171, 1993.

[69] George Casella and Roger L Berger. *Statistical inference*, volume 70. Duxbury Press Belmont, CA, 1990.

[70] Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *JASA*, 46(253):68–78, 1951.

[71] Sebastian Herbert, Siddharth Garg, and Diana Marculescu. Exploiting process variability in voltage/frequency control. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 20(8):1392–1404, 2012.

[72] Jian Li and Jose F Martinez. Dynamic power-performance adaptation of parallel computation on chip multiprocessors. In *High-Performance Computer Architecture, 2006. The Twelfth International Symposium on*, pages 77–87. IEEE, 2006.

[73] A.K. Mishra *et al.* A case for dynamic frequency tuning in on-chip networks. *MICRO*, 2009.

[74] J. Park *et al.* Accurate modeling and calculation of delay and energy overheads of dynamic voltage scaling in modern high-performance microprocessors. *ISLPED*, 2010.

[75] K. Mazumdar *et al.* Breaking the power delivery wall using voltage stacking. *GLSVLSI*, 2012.

[76] Hwisung Jung and Massoud Pedram. Supervised learning based power management for multicore processors. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 29(9):1395–1408, 2010.

[77] Ramazan Bitirgen, Engin Ipek, and Jose F Martinez. Coordinated management of multiple interacting resources in chip multiprocessors: A machine learning approach. In *Proceedings of the 41st annual IEEE/ACM International Symposium on Microarchitecture*, pages 318–329. IEEE Computer Society, 2008.

[78] Yanzhi Wang, Qing Xie, Ahmed Ammari, and Massoud Pedram. Deriving a near-optimal power management policy using model-free reinforcement learning and bayesian classification. In *Proceedings of the 48th Design Automation Conference*, pages 41–46. ACM, 2011.

[79] Gaurav Dhiman and Tajana Simunic Rosing. Dynamic power management using machine learning. In *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, pages 747–754. ACM, 2006.

[80] Ying Tan and Qinru Qiu. A framework of stochastic power management using hidden markov model. In *Proceedings of the conference on Design, automation and test in Europe*, pages 92–97. ACM, 2008.

[81] Li Shang, Li-Shiuan Peh, and Niraj K Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings. The Ninth International Symposium on*, pages 91–102. IEEE, 2003.

[82] Pingqiang Zhou, Jieming Yin, Antonia Zhai, and Sachin S Sapatnekar. Noc frequency scaling with flexible-pipeline routers. In *Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design*, pages 403–408. IEEE Press, 2011.

[83] Hiroki Matsutani, Yuto Hirata, Michihiro Koibuchi, Kimiyoshi Usami, Hiroshi Nakamura, and Hideharu Amano. A multi-vdd dynamic variable-pipeline on-chip router for cmps. In *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, pages 407–412. IEEE, 2012.

[84] Josep Torrellas. Architectures for extremescale computing. *Computer*, 2009.

[85] Ryan Cochran, Can Hankendi, Ayse K Coskun, and Sherief Reda. Pack & cap: adaptive dvfs and thread packing under power caps. In *Proceedings of the 44th annual IEEE/ACM international symposium on microarchitecture*, pages 175–185. ACM, 2011.

[86] Peng Li, Lawrence T Pileggi, Mehdi Asheghi, and Rajit Chandra. Ic thermal simulation and modeling via efficient multigrid-based approaches. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(9):1763–1776, 2006.

[87] Eric GT Bosch. Thermal compact models: An alternative approach. *Components and Packaging Technologies, IEEE Transactions on*, 26(1):173–178, 2003.

[88] Ting-Yuan Wang and CC Chen. 3-d thermal-adi: A linear-time chip level transient thermal simulator. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 21(12):1434–1445, 2002.

[89] Da-Cheng Juan, Yi-Lin Chuang, Diana Marculescu, and Yao-Wen Chang. Statistical thermal modeling and optimization considering leakage power variations. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012*, pages 605–610. IEEE, 2012.

[90] Zhiliang Qian, Da-Cheng Juan, Paul Bogdan, Chi-Ying Tsui, Diana Marculescu, and Radu Marculescu. Svr-noc: A performance analysis tool for network-on-chips using learning-based support vector regression model. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 354–357. EDA Consortium, 2013.