Techniques for Adaptive Power and Thermal Sensing and Management of Multi-core Processors

by
Ryan Cochran

B.Sc., Brown University; Providence, RI, 2008

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in School of Engineering at Brown University

PROVIDENCE, RHODE ISLAND

May 2013

© Copyright 2013 by Ryan Cochran

This dissertation by Ryan Cochran is accepted in its present form
by School of Engineering as satisfying the
dissertation requirement for the degree of Doctor of Philosophy.

Recommended to the Graduate Council

Date________________ ____

Sherief Reda, Advisor

Date________________ ____

Ruth Iris Bahar, Reader

Date________________ ____

Pedro Felzenszwalb, Reader

Approved by the Graduate Council

Date________________ ____

Peter M. Weber, Dean of the Graduate School

Vitae

Ryan Cochran was born in West Chester, PA in 1985. He received his B.Sc. in Electrical Engineering with Honors from Brown University in 2008. He returned to Brown in the fall of 2008 to complete a 1-year Masters program, which quickly grew into a Ph.D. project. His principal research areas include thermal and power modeling, management, and sensing for current and future integrated circuit technologies.

Ryan_Cochran@brown.edu
http://www.scale.engin.brown.edu/~ryan
Brown University, RI, USA

Publications:

1. R. Cochran and S. Reda, "Thermal Prediction and Adaptive Control Through Workload Phase Detection," revision under review for *ACM Transactions on Design Automation of Electronic Systems*, 2012.
2. S. Reda, R. Cochran, and A. Coskun, "Adaptive Power Capping for Servers with Multi-threaded Workloads," to appear in *IEEE Micro Journal*, 2012.
3. S. Reda, A. N. Nowroz, R. Cochran, S. Angelevski, "Post-Silicon Power Mapping Techniques for Integrated Circuits," to appear in *ElSevier VLSI Integration Journal*, 2012.

4. R. Cochran, C. Hankendi, A. Coskun and S. Reda, ”Pack & Cap: Adaptive DVFS and Thread Packing Under Power Caps,” *Proceedings of the International Symposium on Microarchitecture*, pp. 175-185, 2011. Acceptance rate 21%.

5. R. Cochran, C. Hankendi, A. Coskun and S. Reda, ”Identifying the Optimal Energy-Efcient Operating Points of Parallel Workloads,” *Proceedings of the International Conference on Computer-Aided Design*, pp. 608-615, 2011. Acceptance rate 30%.

6. S. Reda, R. Cochran, and A. N. Nowroz, ”Improved Thermal Tracking for Processors Using Hard and Soft Sensor Allocation Techniques,” *IEEE Transactions on Computers*, Vol. 60(6), pp. 841 - 861, 2011. Acceptance rate 23%.

7. R. Cochran, A. N. Nowroz and S. Reda, ”Post-Silicon Power Characterization Using Thermal Infrared Emissions,” *Proceedings of the International Symposium on Low-Power Electronics and Design*, pp. 331-336, 2010. **Best Paper Award.** Acceptance rate 23%.

8. A. N. Nowroz, R. Cochran and S. Reda, ”Thermal Monitoring of Real Processors: Techniques for Sensor Allocation and Full Characterization,” *Proceedings of the Design Automation Conference*, pp. 56 - 61, 2010. Acceptance rate 24%.

9. R. Cochran and S. Reda, ”Consistent Runtime Thermal Prediction and Control Through Workload Phase Detection,” *Proceedings of the Design Automation Conference* , pp. 62 - 67, 2010. Acceptance rate 24%.

10. R. Cochran and S. Reda, ”Spectral Techniques for High-Resolution Thermal Characterization with Limited Sensor Data,” *Proceedings of the Design Automation Conference*, pp. 478 - 483, 2009. Acceptance rate 22%.

Acknowledgements

I would like to express my sincerest gratitude to my advisor, Prof. Sherief Reda, for his patience, encouragement, and the countless hours he has contributed to help make this thesis a reality. Without his constant guidance and insight, this work would not have been possible.

I am also grateful to my committee members, Prof. Iris Bahar and Prof. Pedro Felzenszwal, for generously giving their time to review this thesis and attend my disseration. Their comments and questions are invaluable to this work.

I would like to thank all of my co-authors, including Prof. Ayse Coskun and Can Hankendi from Boston University, my lab mates Monami Nowroz and Stefan Angelevski, and of course Prof. Sherief Reda. I owe much of my output over the past 4 years to their productivity and hard effort.

Thank you to the undergraduates who have contributed so much to the lab infrastructure. Thank you to Patrick Temple and Sriram Jayakumar. Their contributions are the backbone of the graduate research that takes place.

I would like to thank all of my friends and colleagues in the computer engineering laboratory. Thank you for making my extended stay at Brown memorable and fun. Thank you to my fellow lab mates Monami Nowroz, Al-Hussein El-Shafey, Kapil Dev, Kumud Nepal, Onur Ulesel, Roto Li, Dimitra Papagiannopoulou, and Francesco Paterna for sharing in my graduate experience. Thank you to all those who provided a welcome distraction at the foosball table.

Thank you to my housemate Octavian Biris for his moral support in completing this thesis and for the fun times at 111 Governor.

Last but not least, I would like to thank my parents and Leslie for their advice and unwavering support. Everything I have achieved to this point began with what I learned from them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Characterization

In the past decade, power consumption has become the primary factor in overall microprocessor design complexity due to ideal geometric scaling and non-ideal electrical scaling. While geometric scaling permits smaller feature sizes and thus more transistors per silicon-die area as well as faster transition speeds, non-ideal threshold voltage scaling prevents a proportional reduction in power supply voltage for noise and performance reasons. With each process generation, power consumption becomes increasingly constrictive on the performance that can be realized, and it is no longer viable to simply increase the clock speed of existing designs. This barrier to improvement is known as the “power wall”.

Higher power consumption limits processor performance in three ways. First, increased power dissipation leads to increased on-chip power spatial densities and hence higher temperatures. Elevated temperatures cause circuits to deteriorate structurally. All circuit breakdown phenomenon (e.g., electromigration, time dependent dielectric breakdown, and negative bias temperature instability) are highly temperature dependent [89], and thermal cycles create mechanical stresses due to expansions and contractions [11]. It is increasingly difficult to design cost-efficient cooling apparatus capable of removing the heat resulting from chip operation.

Second, increased power dissipation increases the demands on the on-chip and off-chip power delivery networks. It is increasingly difficult to cost-effectively deliver power, particularly in large data centers in which thousands of machines operate in parallel. Traditionally, each data center is given a maximum power constraint and is populated with the maximum number of server units that will stay within this limit. Power constraints also implicitly limit energy consumption.

Finally, higher power dissipation per unit of performance decreases energy efficiency, or the amount of energy necessary to perform a task. In battery-powered devices, cost-effective battery design limits the amount of performance that can be obtained while still maintaining acceptable battery life. In large data-centers, electrical energy billing costs comprise a huge component of the cost of operation. With modern data centers growing larger and denser in order to meet increasing computational demand, energy consumption is fast becoming the largest contributor to the total cost of ownership of data centers and high performance computing (HPC) clusters [31, 4].

These costs taken together limit performance potential. In addition to these restrictions, thermal and power margins must be maintained in order to account for unpredictable workload and environmental conditions. These margins are carefully calibrated to avoid power and thermal emergencies that could degrade performance or damage the system. In order to mitigate the impact of these margins on performance, nearly all modern microprocessors are equipped with “control knobs” that manage the tradeoff between performance and power. In the average case, the processor can maintain a high level of performance without threatening power and thermal margins. In the rare instances in which the power dissipation or temperatures exceed acceptable levels, the processor throttles back performance and reduces power consumption. The techniques for detecting and responding to excess power constitute dynamic power management (DPM), and likewise those that respond to temperature constitute dynamic thermal management (DTM).

There are a number of control "knobs" available for DPM and DTM. The most popular throttling technique is dynamic voltage frequency scaling (DVFS), in which the processor scales back processor frequency and voltage for up to cubic reduction in power. In addition there are a number of architectural throttling techniques in which pipeline resources and functionality are scaled up or down: i-cache throttling, decode throttling, speculation throttling, cache resizing [11, 3]. In all cases, clock gating can be used in order to prune the clock tree in unused resources, thus reducing wasted power consumption.

In addition, performance gains in modern processors are obtained through increased parallelism in hardware and software, which presents new opportunities for DTM and DPM control. Because of the "power wall", it is increasingly difficult to get performance gains by scaling clock speeds. Instead, performance gains are achieved through increased hardware parallelism in the form of multi-core processing. Nearly all computing devices from the embedded to high performance use multi-core architectures. With the increased hardware parallelism in multi-core systems, workloads are becoming increasingly parallel. DTM and DPM techniques use thread management techniques such as thread scheduling and thread migration.

In order for a DTM or DPM technique to make a control decision, it must anticipate the effect that decision will have on the power dissipation, temperature, and performance of the system. The relationship between control decision and these metrics depends on the device architecture and the system configuration, which remain static during runtime.

However, they are also strongly dependent on the workload characteristics. Given a specific architecture operating at a single control setting, the range of power and temperature values vary dramatically within and across workloads. Accounting for only the average power and temperature behavior in response to control decisions yields suboptimal results.

Workload-sensitive power sensing and management poses fewer challenges than for temperature. For one, the value being controlled is typically the total chip power consumption, which can be measured with a single device. Measurements from these devices can be combined with feedback control techniques for robust control. Even in the absence of power telemetry, power can be estimated with relatively simple linear regression models [67, 93]. A chip's power consumption breaks down into two main components: dynamic and static power dissipation. The dynamic power consumption P_{dyn} arises from the transistor and interconnect switching activity according to

$$P_{dyn} = \frac{1}{2}\alpha C_{eff} f {V_{dd}}^2 = k f {V_{dd}}^2, \tag{1.1}$$

where f is the processor frequency, V_{dd} is the power supply voltage, and $k = \frac{1}{2}\alpha C_{eff}$ is a proportional constant that includes the switching activity α and the effective load capacitance C_{eff}. This proportional constant captures the sum of the effective power contributions from each functional unit (e.g. floating point unit, branch prediction unit, cache). Performance counters, which are dedicated hardware registers that count architectural events for various functional units (e.g. floating point operations, branch prediction misses, cache misses), can be used to estimate utilization. Thus, the power consumption can be estimated as a linear combination of the performance counter measurements. Given a value for k, the effect of a DVFS control decision can be projected by direct application of Equation 1.1 by plugging in the desired frequency and supply voltage.

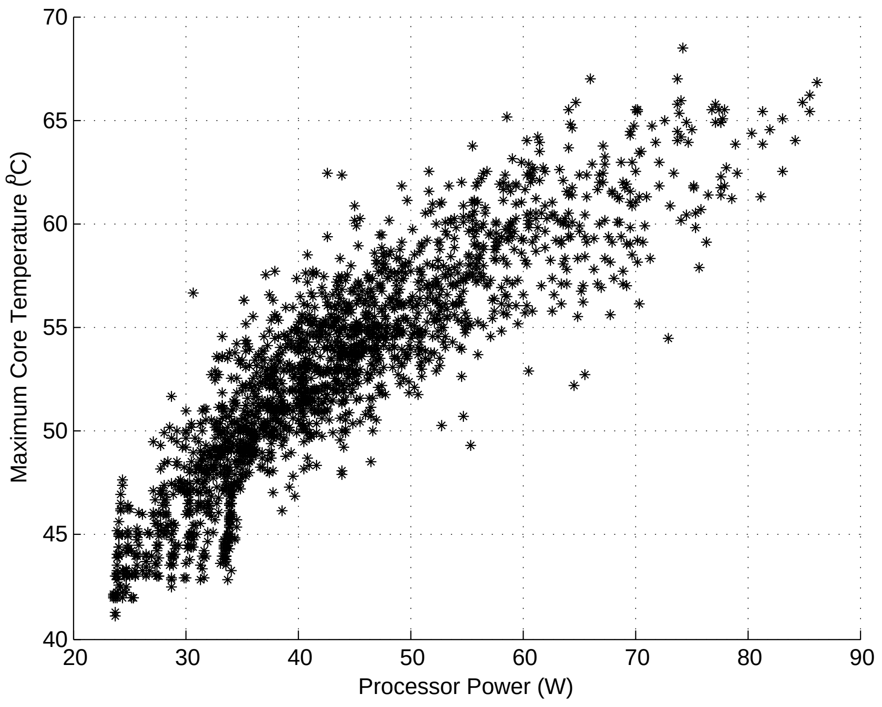

Figure 1.1: Maximum core temperature for a quad-core processor as a function of power consumption.

Adaptive temperature sensing and management is considerably more difficult and requires more sophisticated approaches. The value of interest is the *maximum* thermal hot spot magnitude, which is a function of the power spatial distribution $p(\vec{\mathbf{r}}, t)$ at the 3-D position vector $\vec{\mathbf{r}}$ and time t according to the heat diffusion equation

$$\nabla \cdot (k(\vec{\mathbf{r}})\nabla T(\vec{\mathbf{r}}, t)) - \rho c \frac{\partial T(\vec{\mathbf{r}}, t)}{\partial t} = -p(\vec{\mathbf{r}}, t), \tag{1.2}$$

where $T(\cdot, \cdot)$ represents temperature, ρ is the material density, c is the mass heat capacity, and $k(\cdot)$ is the material thermal conductivity. If thermal sensors did not incur an area overhead, then they could be placed at every possible hot spot location and the device would always have an accurate thermal assessment. In reality, however, sensors occupy a significant die area and complicate routing. Thus, they must be placed judiciously, and it is possible for hot spots to occur at non-sensor location. As evidenced in Figure 1.1, total processor power consumption is a poor proxy for maximum temperature, as two workloads may have identical power consumption with very different maximum die temperatures. The linear regression model does not take into account the nonlinearities that arise from shifting power densities. For instance, if the maximum temperature hot spot occurs at the floating-point unit, it will be a strong function of the number of floating point operations. If the maximum hot spot happens at a distant location, however, its magnitude will be nearly independent of the floating point activity.

The most detailed models used for runtime temperature control modeling in the literature use the finite difference approximation for the heat equation

$$\mathbf{T}(t+h) = h\mathbf{C}^{-1}\Big(\mathbf{P}u(t) - \mathbf{A}\mathbf{T}(t)\Big) + \mathbf{T}(t), \tag{1.3}$$

where h is the time step size, t is time, $\mathbf{T}$ and $\mathbf{P}$ are $N \times 1$ are vectors of temperatures and power dissipations defined for a set of N locations, $\mathbf{A}$ and $\mathbf{C}$ are $N \times N$ matrices of thermal conductivities and capacitances, and $u(t)$ is a unit step function [11, 105]. Estimating $\mathbf{A}$ and $\mathbf{C}$ requires knowledge of the processor floorplan and estimating $\mathbf{P}$ requires numerous assumptions about the relationship between workload behavior and local power dissipation, all of which must be manually validated. Without fine-grained power measurements for a real processor, these models are difficult to validate. Any manual validation performed for one chip design does not extend to future designs in which the physical layout and functional unit power characteristics change significantly. In addition, the exponential dependence of leakage power consumption on temperature must be accounted for in $\mathbf{P}$.

1.2 Contributions

In addressing the challenges inherent to adaptive, workload-sensitive DPM and DTM, this thesis makes the following major contributions:

- **Power and Thermal Sensing:** We develop infrastructure for measuring power consumption and die temperatures of real processors with a state-of-the-art thermal infrared camera using oil cooling [94, 19, 21]. Based on the characterization results from the camera, we propose a formulation for thermal sensor allocation to minimize the thermal tracking error during runtime. We prove that our formulation leads to a NP-hard problem and accordingly we propose a heuristic solution method that is composed of constructive and iterative phases. The experimental results show that our method significantly improves upon previous methods in the literature. To circumvent limitations on thermal sensor placement, we proposed two soft sensing techniques that combine the measurements of hard sensors to estimate the temperatures at the ideal locations. Our first technique leverages *a priori* design-time characterization data to seek customized weighted combinations to estimate the temperatures at any desired location. Our second technique uses spectral signal reconstruction techniques based on Fourier analysis for interpolating die temperatures from a limited number of thermal sensors [86, 21]. This soft-sensing technique is designed for cases in which there is no *a prior* thermal characterization through simulations or infrared camera. We show that both soft-sensing techniques significantly improve the thermal estimation accuracy.

- **Power Management:** We introduce thread reduction for multithreaded workloads as a means of meeting low power budgets on a single server node, thus increasing power allocation flexibility for datacenters composed of many server nodes. Because thread reduction is not easily performed dynamically during runtime, we propose *thread packing*, in which multi-threaded workloads are packed onto a variable number of active cores, as a proxy for thread reduction [93, 18, 17]. We show that thread reduction and thread packing have nearly identical power and performance characteristics. We then devise a novel DPM method, *Pack & Cap*, which makes optimal DVFS and thread packing control decisions such that performance is maximized within a power budget. We illuminate power and performance tradeoffs between thread packing and DVFS and develop heuristics for navigating Pareto efficient control settings. We implement Pack & Cap with several candidate power models, including a multi-gain feedback controller as well as open-loop and closed-loop linear regression models. We compare these techniques to a baseline feedback technique and demonstrate up to 10% reduction in average runtime and up to 53% improvement in average power cap accuracy. We also compare to our previous work in [18], which uses a multinomial logistic regression (MLR) classifier, and show up to 30% runtime improvement with similar power cap accuracy.

- **Thermal Management:** We introduce a novel technique for estimating the temperature control response that uses the concept of workload phases [20, 22]. We use classification techniques from machine learning to classify workload execution intervals into phases as a function of performance counter measurements. We then associated a thermal model with each workload phase. This method addresses the inherent nonlinearities of workload-sensitive temperature modeling on multi-core systems, and we demonstrate improved accuracy over the linear regression model. Although this model can be applied to many of the control objectives in the literature, we demonstrate this technique in the case of proactive DVFS thermal control. We compare to state-of-the-art model predictive control (MPC) techniques in the literature and show 5.8% improvement in instruction throughput with the same number of thermal violations. When we compare our thermal phase classification techniques to the sequential-probability-ratio-tests (SPRT) used for switching thermal models in previous works, we demonstrate a 2.9% improvement in instruction throughput and an 97% reduction in thermal violations. In comparison to an optimally tuned proportional-integral (PI) feedback control technique, we improve instruction throughput by 3.9% with a 94% reduction in the number of thermal violations.

The remainder of this thesis is structured as follows. Chapter 2 surveys the state-of-the-art sensing and management techniques in the literature. Chapter 3 details our thermal infrared camera setup and presents our techniques for thermal sensor placement and soft-sensing. In Chapter 4, we introduce the Pack & Cap methodology for maximizing performance of multithreaded workloads within a power budget in a server environment. In Chapter 5, we describe a technique for thermal control modeling using workload phases and apply it to proactive DVFS control. Finally, in Chapter 6, we summarize our findings and outline directions for future research.

Chapter 2

Background

2.1 Power and Thermal Sensing

The main difficulty in hot spot tracking of modern processors is that there can be potentially large variations in the spatial and temporal die temperatures. These variations arise from a number of causes:

- Advanced processors with larger die area naturally exhibit more spatial thermal variations.
- Workloads have different power consumption profiles and different resource utilizations. For example, a floating-point workload will have its hot spots in locations that are different from an integer-based workload. The wide variations in the workloads of general-purpose processors lead to larger spatial and temporal thermal variations.
- Current and future trends of processor organization are moving towards many-core architectures with potentially 100s–1000s of cores [10]. Many-core processors will

further localize the power density, thus increasing the number of potential hot spot locations [46].

Given the gravity of thermal hot spots and the variations in their locations, processor designers utilize thermal sensors that track the processor's hot spot temperatures. Dynamic thermal management (DTM) techniques utilize these sensor measurements to dynamically adapt the chip's performance depending on its hot spot temperature. Methods of adapting performance include frequency and voltage scaling [28, 82, 64, 41, 85, 58, 60, 114, 123]; adjusting throughput through the control of issue width and branching speculation [12, 104, 57, 60, 3]; and operating system techniques such as thread/computation migration and scheduling [43, 90, 28, 121, 24, 26, 114, 116].

To enable DTM effectiveness, it is necessary to supply the DTM controller with thermal sensor measurements at all possible hot spot locations. However, thermal sensors and their support circuitry utilize die area real estate and increase design complexity. For example, surveying some commercial-grade digital thermal sensors at `design-reuse.com`, we found that a digital thermal sensor requires an area of 0.25 mm^2 in 180 nm technology and an area of 0.10 mm^2 in a 65 nm process. For digital thermal sensors, the voltage signal of a thermal diode is routed to and measured by an analog to digital converter (ADC) that stores its results in a register that is periodically checked by the DTM system. Digital thermal sensors consume a good portion of die area mainly due to the need to accommodate the ADC. Because die size is the main recurring cost during fabrication, there is an inherent trade-off in thermal monitoring. On one hand, designers would like to reduce costs by using the fewest number of thermal sensors, while on the other hand, the gravity of runtime thermal problems require higher thermal resolution. Greater thermal sensing error forces DTM controllers to use more conservative thermal margins, thus constraining performance unnecessarily.

To address the problem of thermal sensor allocation in processors, a number of techniques were proposed in the literature [70, 83, 84, 98, 78, 56, 68, 122]. Rotem *et al.* demonstrated that inaccuracies in thermal tracking decreases the processor's performance and wastes power [98]. In particular, it was shown that a 1°C accuracy translates to 2 W power savings, and that due to lack of proximity, sensor measurements and hot spot temperatures could differ by up to 10°C. Lee *et al.* propose the use of thermal simulators to identify the locations of hot spots using different workloads [71] . Mukherjee and Memik describe a clustering algorithm that computes the thermal sensor positions that best serve clusters of potential hot spot locations [83, 78]. The locations of these hot spots are identified via workload thermal simulation, and the clusters are computed using a modified k-means algorithm that incorporates the spatial locations of the hot spots and their temperatures. Even with good placement, it is likely that sensors will fail to detect hot spots when the number of sensors are far less than the number of hot spot locations. Thus, Long *et al.* advocate using a grid-based interpolation scheme for chip multiprocessor [75]. To compensate for the inherit limitations in regular allocation, the hot spot temperature is estimated by interpolating the measurements of its immediate sensor neighbors in the grid. To circumvent the restrictions on the number of thermal sensors, another research direction proposes estimating the hot spot temperatures where no sensors are embedded using frequency-domain interpolation techniques [21, 86]. Liu [74] proposes using *Kriging* estimation as a general framework for estimating variability (whether manufacturing, thermal or IR drop) at various chip locations during design time. A Kriging temperature estimator models the die temperature as a random field and computes unknown temperatures using a *variogram* function, which captures spatial correlations as a function of distance. Given the inherent noise arising in thermal diodes, a number of papers provide techniques to reduce the noise and provide accurate thermal tracking [56, 122].

Reconfigurable computing gives an excellent balance between performance and flexibility. In a reconfigurable chip, hot spot locations depend on the configured architecture which is not known at the design and manufacture time of the FPGA. Thus, there is no way of knowing *a priori* the best locations for the sensors. A number of papers [84, 68] propose thermal sensor allocation algorithms for FPGAs where the FPGA's reconfigurable area is covered with a number of sensors with each sensor monitoring a range of thermally-correlated configurable logic blocks.

High-end chips, such as multi-core processors and graphical processor units, are equipped with integrated thermal sensors that monitor the on-chip temperatures during runtime. Thermal diodes, which translate temperature variations into voltage variations, are a popular choice for temperature sensing. The diode voltage signal is routed to and measured by an analog to digital (A/D) converter. In older processor technologies, the A/D conversion takes place on the board, while in newer processors (e.g., Intel's Core i7), the A/D conversion takes place on-chip. State-of-the-art chips are typically equipped with more than one thermal sensor. There are a number of reasons for this: (1) complex chips with large die area require more thermal sensors to capture temperatures at a wide range of locations; (2) the unpredictability of a processor's workload can lead to continuous migration of hot spots; and (3) within-die manufacturing variations lead to leakage variability that can further conceal the locations of the thermal hot spots. The thermal sensors, together with their support circuitry and wiring, complicate the design process and increase the total die area and manufacturing costs. Thus, there is an inherent tradeoff in thermal monitoring. On the one hand, designers would like to reduce costs by using the fewest number of thermal sensors, while on the other hand, the gravity of runtime thermal problems require higher thermal resolution.

2.2 Power Management

Most modern processors eliminate unnecessary static and dynamic power by switching unused components to low-power idle states and by clock gating. Clock gating is a technique in which additional logic is used to disable portions of the system clock tree, thus preventing switching activity on unused components. These low-power states are implemented in hardware, but their functionality is exposed to the software level via well-known interfaces, such as the Advanced Configuration and Power Interface (ACPI) [1]. There is typically a tradeoff between idle state power savings and the time required to return to being active. In order to speed up these transitions Meisner *et. al* [77] introduce the PowerNap server architecture, in which the entire server rack transitions between active and idle state rapidly in response to instantaneous load. These techniques reduce fixed system power costs that do not vary with processor activity (disk power, external memory power, etc.). Within an active state, a processor can dynamically control power consumption using dynamic voltage frequency scaling (DVFS). This is a popular technique for dynamic control because changes in frequency and voltage can be performed with minimal overhead (on the order of micro-seconds), and it provides fine-grained control of the tradeoff between power and performance. As indicated by Equation 1.1, the dynamic power is linear with frequency and quadratic with supply voltage. Given that the supply voltage scales somewhere between linearly and quadratically with the frequency, reductions in processor frequency can yield up to cubic reductions in power consumption.

As described in Chapter 1, these control knobs allow dynamic power management (DPM) techniques to manage the tradeoff between performance and power under dynamic workload and environmental conditions. The control objective for DPM comes in two forms. In the first form, the DPM controller attempts to maximize performance within a power budget. This objective is appropriate for applications in which faster is always considered better as long as the constraint is met. Various techniques differ in terms of how stringently the budget is enforced. For instance, Kontorinis *et. al* use tables and centralized control hardware to guarantee that no combination of chip resource can be fully activated simultaneously such that the power budget is violated [62]. This stringent enforcement of peak power constitutes *hard power capping*, in which the explicit goal is to never violate the power budget. On the other end of the spectrum, many works employ *soft power capping*, in which the power budget can be exceeded so long as the average power consumption is within the budget. Gandhi *et. al* meet power budgets by rapidly inserting idle cycles during execution such that an average power consumption is maintained [35].

In the second form, the DPM controller attempts to maximize energy efficiency (energy consumed per unit of work) with a lower-bound on performance. In these cases, faster is not better if the increase in power degrades energy efficiency. Many works perform profitability estimation, in which the controller weighs the potential improvement in performance against the expected increase in power. Several works use performance counter measurements in order to perform profitability estimation for DVFS [30, 27]. Other works use offline workload characterization in order to determine the optimal DVFS settings [76, 18, 17].

Multi-core processors introduce new opportunities for power management by enabling additional degrees freedom, including per-core DVFS. Some commercial multi-core processors support independent frequencies for each core, but lack per-core voltage domains. Independent voltages require extensive design investments in the power-delivery network and the off-chip power regulators. Although some works suggest that the performance benefits of per-core DVFS do not outweigh the added complexity of multiple voltage and frequency domains [44], there are numerous works that develop DPM techniques for per-core DVFS [50, 79, 112, 6, 100, 99].

Multi-core processors also allow additional degrees of freedom in terms of thread scheduling and allocation. Rangan *et al.* propose an alternative to per-core DVFS in which the power level of each core is fixed, and threads are rapidly migrated among these cores in order to adapt to the time-varying computation needs of each thread [92]. Multithreaded workloads that take advantage of the hardware parallelism in multi-core architectures are increasingly prevalent and present additional challenges to control. Several works explicitly account for the interdependence among threads in multi-threaded workloads. Several works propose a power balancing strategy that dynamically adapts the per-core power budgets based on thread criticality [14, 8].

When considering DPM for large data-centers, which are composed of potentially thousands of individual machines, the control objectives we have defined for individual machines are considered at an aggregate level. Data-center power management presents unique challenges because individual server nodes incur relatively large idle power consumptions when active, and the typical data-center utilization is usually below the maximum capacity. As a result, much of the overall power consumption is wasted on machines that are doing very little work. In addition to maintaining power within the limits of the power delivery network, data-center DPM techniques aim for *energy-proportional* computing, in which the overall system power consumption scales linearly with the overall utilization [4]. A number of previous works deal with energy-proportional computing at the aggregate level using existing hardware-level control knobs on individual machines. Tolia *et. al* investigate techniques for using off-the-shelf hardware in order to approximate energy proportional computing at the aggregate level [108]. Similarly, Elnozahy *et. al* use dynamic voltage scaling (DVS) and rapid switching between on and off states in order to reduce power consumption of web servers during periods of low utilization [29]. Gandhi *et. al* develop a theoretical model for power allocation in server farms and show that it is often advantageous for energy efficiency to operate servers at a lower-power states [34].

2.3 Thermal Management

The thermal control response depends on shifting spatial power densities which are a complex function of workload behavior, and yet it is impossible to anticipate every possible workload configuration when designing a DTM technique. In order to guarantee thermal safety for the full range of thermal behaviors, many commercial processors combine thermal sensor measurements with reactive feedback techniques [7]. If the measured temperature exceeds a threshold, the system reacts by scaling back power consumption with the available control knobs. In recent academic works on DTM, it is commonplace to augment thermal sensor information with a thermal model that anticipates the thermal trajectory. These techniques generally fit a mathematical model to a local window of observed temperatures arising from the execution of an application. After being learned, a thermal model is used to extrapolate future temperatures. The model coefficients are typically estimated to give the total least square errors between the model results and the observed temperatures in a set of training samples. By increasing the certainty about the thermal control response, the system can avoid thermal violations and associated performance penalties.

Many these previous works do not model the future temperature as an explicit function of the power consumption, but instead model stationary patterns in the temperature time series with auto-regressive (AR) and auto-regressive moving average (ARMA) models [25, 23, 118]. The temperatures projected by these models are used in place of the thermal sensor readings to perform control. Changes in the temperature behavior that result from changing workload characteristics are handled using online update techniques. One approach is to develop a bank of ARMA models and use a sequential-probability ratio test (SPRT) to determine the likelihood of each model relative to each alternative model given the latest thermal observations [25, 23]. The result of a statistical hypothesis test is used to select the likeliest model, or if no model aptly describes the data, a new model is generated online using the latest observations. Other works use recursive least-squares update techniques to efficiently re-estimate AR coefficients online using the latest observations [118].

One drawback of these approaches for DVFS control is that the models are not explicit functions of voltage or frequency. The outputs are static as a function of DVFS control decisions and cannot be queried for the optimal setting. In addition, there is a dependence between the model parameters and the control decisions that can lead to unpredictable behavior. In the case of DVFS, the optimal coefficients in the AR or ARMA model depend on previous history of voltages and frequencies, and are used to inform future voltage and frequency decisions. As a result, the sequence of future control decisions is an unpredictable function of previous decisions. These techniques must rely on online update techniques to capture changing workload characteristics. The computational cost of matrix inversion in relearning model coefficients offsets the benefits of performing predictive control. We avoid this computational cost by learning our models offline with training data that spans a large range of workload behaviors. We then validate our models at runtime using workload mixtures unseen in the training data.

Other works model the temperature as an explicit function of workload characteristics without using thermal sensor measurements. Several works use a linear combination performance counters and core utilization metrics to estimate the temperature [63, 69]. These works are useful as low-cost alternatives to thermal simulation techniques when there are limited or no thermal sensor measurements. However, they do not account for the transient nature of temperature or thermal coupling, and most modern high-performance processors are equipped with per-core thermal sensors.

The discrete time-invariant state-space model in Equation 2.1 is a standard model used for estimating temperature in the literature [11, 112, 102, 120, 65, 109].

$$T_m[i+1] = \sum_{n=1}^{N} a_{mn} T_n[i] + a_m^{idle} + \widehat{g}\Big(s,\ \ \mathbf{x_m}[i]\Big) \tag{2.1}$$

In this model, the temperature for node m at time $i+1$ is a linear combination of its own temperature and the temperature of the other $N-1$ nodes on the previous time interval as well as the workload thermal contribution $\widehat{g}(\cdot,\cdot)$ and the idle thermal contribution a_m^{idle}. The workload thermal contribution is a function of the system control setting s as well as a vector of workload metrics $\mathbf{x_m}$ for core m. It is a natural choice for modeling temperature because it approximates continuous first-order differential equations that govern transient thermal behavior [11, 112, 102, 120, 65, 109, 22]. It explicitly models thermal time constants and thermal coupling among thermal nodes with the coefficients a_{mn}. It also explicitly models the workload thermal contribution $\widehat{g}(\cdot,\cdot)$ as a function of the control setting and workload characteristics. The approaches in the literature differ in the form of the voltage, frequency, and workload dependence. Several studies use the relationship $\widehat{g}(\cdot,\cdot) \propto f^{\alpha}$, where α is a parameter between 1 and 2 depending on how the voltage scales with frequency [117]. Other studies relate $\widehat{g}(\cdot,\cdot)$ to frequency f, supply voltage V_{dd}, and workload cycles-per instruction (CPI) using the empirical formula

$$\widehat{g}(\cdot,\cdot) = k_a {V_{dd}}^2 f + k_b + \Big(k_c + k_d f\Big) CPI[i]^{k_e}, \tag{2.2}$$

where k_a, k_b, k_c, k_d, and k_e are parameters that must be learned [5, 6]. This formula has a benefit in dynamic frequency scaling (DFS) in that the power is a linear function of frequency (assuming constant voltage). As a result, it can be easily incorporated into a model predictive controller (MPC) solution. The MPC minimizes a quadratic cost function while preventing thermal overshoot for a finite prediction horizon [5, 6, 112].

However, we have found that assuming a functional form for the relationship between power and DVFS setting is a significant source of error. In real processors, the voltage does not always scale with the frequency in a straightforward manner. The voltage may change with frequency for a subset of the DVFS settings, and remain constant for other subsets. Assuming a fixed exponent for f can introduce significant error in this case. To counter this, we calculate a distinct value for $\widehat{g}(\cdot,\cdot)$ for each DVFS setting s.

In a similar vein, we contend that assumptions about the relationship between $\widehat{g}(\cdot,\cdot)$ and workload introduce significant errors. For instance, assuming that the activity factor for a functional unit is a linear function of a performance counter metric ignores changes in power density and hot spot location that may accompany different degrees of utilization. Verifying model assumptions requires detailed floorplan knowledge and a carefully calibrated set of micro-benchmarks that exercise functional units individually in order to learn $\widehat{g}(\cdot,\cdot)$ for each unit at each DVFS setting.

Workload phase classification is a technique for representing the average behavior of a workload over a local time window [55, 103, 51, 15]. It is well established that workloads execute in consistent and repetitive patterns, and this fact is reflected in all manifestations of workload behavior (computational operations, memory operations, power, temperature etc.). These behaviors can change instantaneously and dramatically within a workload or across a mixture of workload. Thus, the average behavior of an entire workload gives an incomplete picture. A *workload phase* is defined as a period of workload execution that exhibits a consistent behavior with respect to a set of metrics.

There is a significant body of literature dealing with phase identification [103, 39, 51, 52, 15, 40, 22]. The problem generally reduces to a classification problem that takes a vector of metrics as input. The metrics used differ depending on the application. Several works [52, 103, 39] define these inputs using traversal counts on basic block vectors (BBV). Each BBV represents code blocks with a single entry and exit point, and the traversal counts reflect the control paths taken within the program. This work in phase classification has yielded SimPoint [40], which finds representative simulation points for workloads with long execution times. The drawback in using the BBV for phase classification is that it requires extensive knowledge of the workload code structure, and phases can only be defined per-workload.

A number works on workload phase identification use metrics derived from performance counters instead [51, 22]. Most commercial processors are equipped with special registers that count various architectural events (μ-ops retired, cache-misses, etc.). The choice of metrics depends on the application. Isci *et. al* [51] use the ratio of memory transaction to μ-ops retired as a measure of application memory-boundedness in order to calculate power phases. In [52], Isci *et. al* use a set of 15 performance counters that measure computational, bus, memory, and branch activity in the context of dynamic power management. Cho *et. al* [15] use instructions-per-cycle (IPC) as the sole metric in order to demonstrate multi-resolution phase analysis with wavelet transforms. Kim *et. al* [61] uses the joint distribution of computational and stall time performance metrics for energy-efficient DVFS control. Our own work [22] on phase-aware temperature prediction uses instructions retired, floating point operations, and conditional branch instructions after dimensionality reduction via principal component analysis.

Given a set of training data vector, the classifier is built using some metric of similarity among all the training vector instances. A common technique in the literature is K-means clustering [103, 39, 40, 15, 22]. The workload is divided into time-slices, each of which is associated with a d-dimensional vector measuring various behaviors within that interval. The K-means clustering method iteratively seeks a set of K d-dimensinoal centroids that minimize the average distance between the input vector and the closest centroid. Isci *et. al* [51] use a simple thresholding scheme on a measure of the application's memory-boundedness in order to define 6 phases. In [52], Isci *et. al* explore other clustering techniques in first pivot clustering and agglomerative clustering. In first pivot clustering, each new sample is compared with a distance metric to an existing set of pivots, and if the measured distance exceeds a particular threshold, then the sample is added as a new pivot to the existing set. Agglomerative clustering is a bottom-up iterative approach. The algorithm begins with N clusters corresponding to N training samples, and at each iteration it performs pairwise comparisons between the clusters with a linkage function and selects the best candidate pair of clusters to combine into a single cluster.

There are other works on thermal prediction that implicitly incorporate workload phases, even if the words "workload phase" do not appear in the text. The approach in [26, 23] defines a set of thermal predictions models which can be interpreted to correspond to workload phases. Transitions between these phases and their corresponding models are detected using statistical hypothesis tests on the measured data for each model. Phase transitions are instead detected when the output errors produced by an alternate model are more statistically likely to occur than those produced by the current model. This likelihood calculation comes in the form of a per-core sequential probability ratio test (SPRT), which uses a log likelihood ratio of the observed data for an alternate phase $k2$ and that of the current phase $k1$. The SPRT method is a form of hypothesis testing for sequential data and is a function of the evidence accumulated from the previous N samples. Acceptance of the phase $k1$ forms the null hypothesis, while acceptance of $k2$ constitutes the alternate hypothesis. For each incoming data sample, this test is performed between the current model and each of the other $K - 1$ candidate thermal models. Acceptance of the null or alternate hypothesis is determined by applying a simple threshold scheme to the log likelihood ratio. Calculation of the log-ratio proceeds so long as its value falls between two user-specified limits $A < B$. If the value is less than the lower limit A, then the null hypothesis model $k1$ is accepted as the true model and the calculation is reset. Likewise, if the value exceeds the upper limit B, then the alternate hypothesis model $k2$ is accepted and the calculation is reset.

Chapter 3

Power and Thermal Sensing Techniques

To manage runtime power and temperatures, it is necessary to use power and thermal sensors. The measurements of the sensors are used by the DPM and DTM systems to adapt the performance, power consumption, and the temperatures given the operational constraints [90, 104, 121, 58]. Power sensing is relatively easier than thermal sensing as it only requires lumped measurement of the system's or the processor's total power. In contrast, the spatial and temporal fluctuations in thermal hot spots make the task of tracking thermal sensing particularly challenging. Furthermore, many-core processors localize power consumption in potentially more than one spot, which further increases the difficulty of thermal sensing [95].

The objectives of this chapter are to provide novel accurate methods for power and thermal sensing and tracking. The contributions are as follows:

- We develop a realistic power sensing and thermal imaging setup to characterize the thermal behavior of real processors during runtime. To tackle the challenges in working with real chips, we devise multiple experimental techniques through the use of oil-based infrared-transparent heat sinks and novel techniques for thermal calibration. We execute a large collection of workloads in different configurations on the experimented processor, and we track the locations of hot spots through space and time. Our characterization results provide valuable insights into the extent of hot spot variations during runtime.

- To track hot spots accurately as they vary depending on workloads and active cores, we propose a thermal sensor allocation algorithm that finds the best sensor locations for a given number of thermal sensors. Our allocation algorithm seeks to track hot spots accurately by minimizing the worst case error between the maximum measurements of the sensors and the hot spot temperatures.
- We propose *soft* sensing computation techniques that use the measurements of the thermal sensors to optimally compute the temperatures where no sensors are embedded. Our soft sensor techniques can improve the thermal tracking resolution and circumvent design constraints on sensor placement. Soft sensors can also substitute for hard sensors, thus reducing the demand on die area. We develop two techniques for soft sensing. One technique is geared for situations where thermal characterization data is available *a priori* from either simulations or infrared imaging. The second technique is suitable for cases where no *a priori* characterization is available.

The remainder of the chapter is organized as follows. Section 3.1 details our experimental power and thermal sensing techniques. It includes a detailed description of new infrared imaging experimental techniques required for thermal imaging of real processors. In Section 3.2, we introduce techniques for thermal sensor allocation given the potential hot spots in the design. In Section 3.3 we describe our soft sensing techniques. In Section 3.4 we summarize the main results of this chapter.

3.1 Experimental Techniques for Sensing

Power Sensing. To develop any lumped power-related model, it is necessary to first collect a large volume of power characterization data. There are generally two approaches used to measure the total electrical current consumption of a computing system. In the first approach, the power supply lines are intercepted and a *shunt resistor* (e.g,. Figure 3.1.a) is inserted in series with the positive supply line. In contrast to regular resistors, shunt resistors have very low resistance (e.g., 1 mΩ) with high accuracy of about $\pm 0.1\%$. The low resistance is needed to avoid adding a voltage drop along the supply line. The changes in voltage across the shunt resistor are proportional to the electrical current variations as dictated by Ohm's law. The second approach uses *clamp meters* (e.g., Figure 3.1.b), which utilize the Hall effect to detect electric current variations in the supply line by measuring the induced magnetic field variations surrounding the supply wire. Clamp meters are less intrusive, but they are less accurate and their measurements tend to be noisy compared to shunt resistors. In both approaches, a digital multimeter or an analog-to-digital device is required to log the measurements of the shunt resistor or clamp meter into the power management system of the computing device.

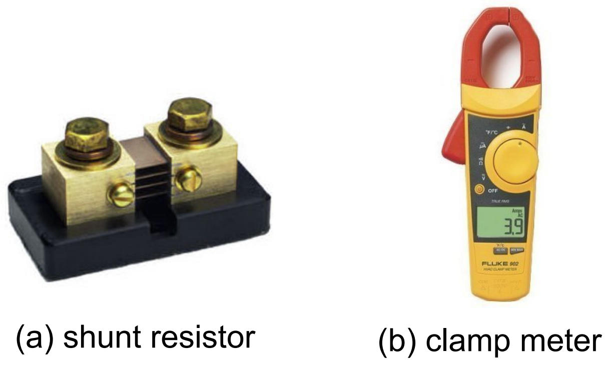

Figure 3.1: Power measurement techniques used for lumped current measurements.

Thermal Sensing. Due to its numerous advantage, flip-chip packaging is the state-of-the-art method in soldering processor chips to external circuitry. In flip-chip packaging, solder bumps are deposited onto the die pads at the top side of the die. To connect the die to external circuitry, the die is flipped over and soldered to the package substrate. By removing the package's heat spreader, one can obtain optical access to every device on the die through the silicon backside. Silicon is transparent in the infrared spectral region (wavelengths longer than 1.1 μm), and this transparency allows the capturing of thermal infrared emissions using infrared imaging techniques.

The main components of a high-sensitivity infrared camera are (1) a focal plane array of photon detectors, (2) a cooler for the array, (3) a lens system, and (4) analog to digital electronics for readout [97]. Photon detectors convert photons with energy greater than their band gap into electron-hole pairs that can be collected by external circuitry. For mid-range infrared imaging, it is necessary to cool the detector array to cryogenic temperatures in order to reduce dark noise. During operation mode, one typically takes an image of the entire die or some particular field of view within the die using the camera after integrating the detected photocurrent for a period of time (at least a few micro to milli seconds) that depends on the emission signal strength.

In our thermal imaging setup we use a FLIR SC5600 infrared camera with a mid-wave infrared spectral range of 2.5 μm – 5.1 μm. Undoped and lightly doped silicon are transparent at the mid-wave infrared range. The camera has 640 $\times$ 512 InSb quantum detectors with 15 μm pitch between detectors. The camera's detectors are chilled to 77 K (-196°C) and have measurement errors less than 20 mK. We use a $0.5\times$ microscopy kit and operate the camera with a frame rate of 100 Hz. Thus, the camera can update its measurements every 10 ms. It is possible that a high-frequency spike could be missed by the camera but because of the thermal capacitances associated with the chip, it is unlikely that such spikes will be significant. To capture the infrared emissions from the back side of

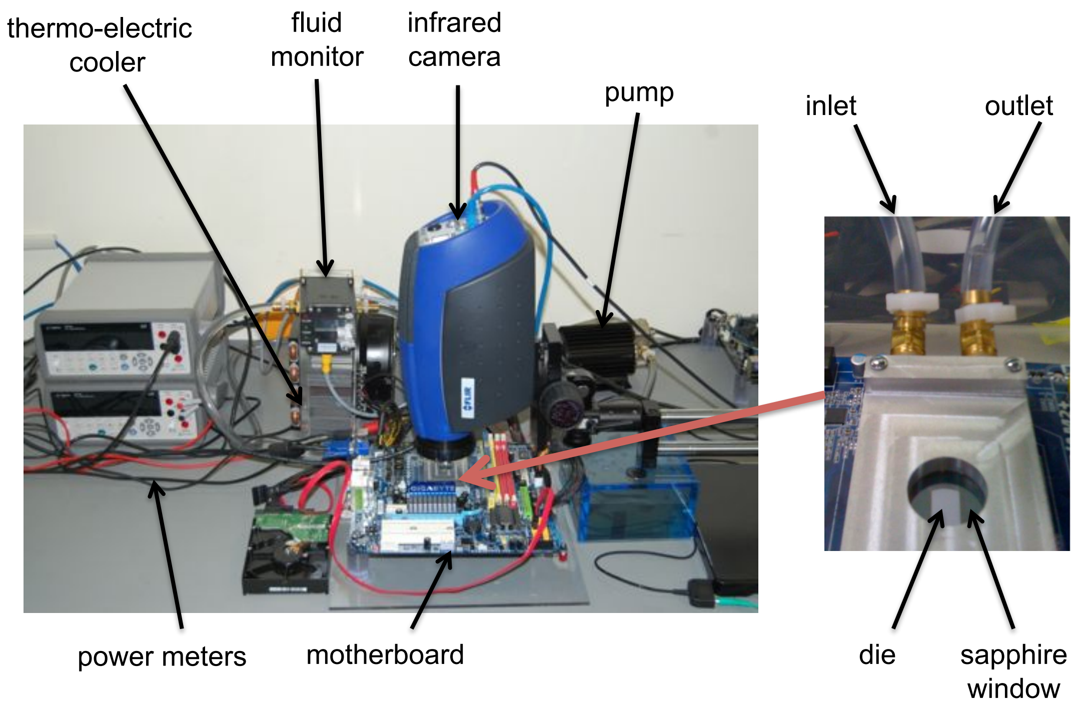

Figure 3.2: Image of our experimental setup.

a processor's die, the heat spreader of the experimented processor is removed. To enable normal operation of the processor and the entire motherboard, it is necessary to remove the heat generated during operation. We machined an infrared-transparent heat sink with a sapphire window that has a 1 mm clearance between the window and the processor die as illustrated in Figure 3.2. Chilled oil at approximately 15°C is pushed through the inlet of the heat sink to form a thin film on top of the die that removes the heat. The oil is pumped continuously at about 1.5 gpm using an external DC pump. The temperature of the oil is controlled using a thermoelectric cooler. A Proteus Fluid Vision 4000 device is used to monitor the oil's flow rate, temperature and pressure just before the oil's entry to the machined sink. The measurements from the flow monitor are logged into a second monitoring computer through a National instrument A/D acquisition device. Monitoring and controlling the temperature of the oil is necessary for thermal calibration as will be explained in the next paragraphs.

Measuring the temperature is complicated by that fact that an infrared camera is really a photon detector that measures the infrared radiation intensities at different parts of the chip. Thus, it is necessary to convert the digital levels (which reflect photon intensities) recorded by the camera to temperatures. This conversion has to take place on a pixel-by-pixel basis for the following two reasons.

1. Radiation intensity is not constant among different materials even if they are at the same temperature. Perfect radiation emitters are *black bodies* with an *emissivity* of 1. The emissions of real materials are a fraction of the black-body level, and each material is characterized with an emissivity value, which is defined as the ratio of that material's thermal emission to that of a perfect black-body at the same temperature [97]. As integrated circuits are composed of different materials (e.g., copper, silicon and dielectrics) with different spatial densities, the radiation intensities of different parts of the chip could be different even if the chip is held at an isothermal temperature by external means.

2. In addition to emissivities, different materials reflect radiation from the surrounding environment with varying intensities. The reflections from the materials of the integrated circuit further obscure the true thermal status of the materials under observation.

To compute the pixel-by-pixel relationships between temperatures and digital levels we devise a calibration method. While the processor is turned off, it is forced to a known isothermal status and the digital levels at every pixel are recorded. The chip's temperature is scaled up at 5°C increments and the digital levels are repeatedly recorded. For example, Figure 3.3 shows the digital levels for two pixels at different temperatures during calibration. To force the chip into an known isothermal status, we devise a computer-based

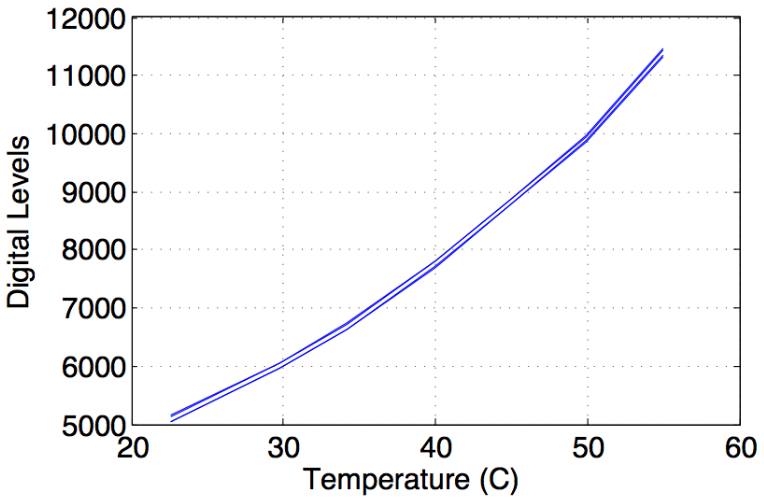

Figure 3.3: Temperatures and their corresponding digital levels for two pixel locations on the die.

feedback control system that controls the thermoelectric cooling/heating capacity through a programmable current supply. The temperature output from the fluid flow monitor is fed into a computer using the A/D acquisition device, and the computer adjusts the voltage supply of the thermoelectric cooler/heater until fluid temperature stabilizes at the desired level.

The relationship between the digital level D_j and temperature t_j of a pixel j can be modeled by an exponential function $D_j = \alpha_j e^{\beta_j t_j}$, where α_j and β_j are per pixel coefficients. This relationship arises from the physics of photon detectors, in which the current of an infrared-sensitive diode depends exponentially on the incident radiation [45]. Using the calibration data, we compute the α_j's and β_j's of every pixel using standard curve fitting techniques. During normal operation, the measured digital levels together with the calibrated α's and β's are used to compute the pixel-by-pixel temperatures according to the exponential relationship.

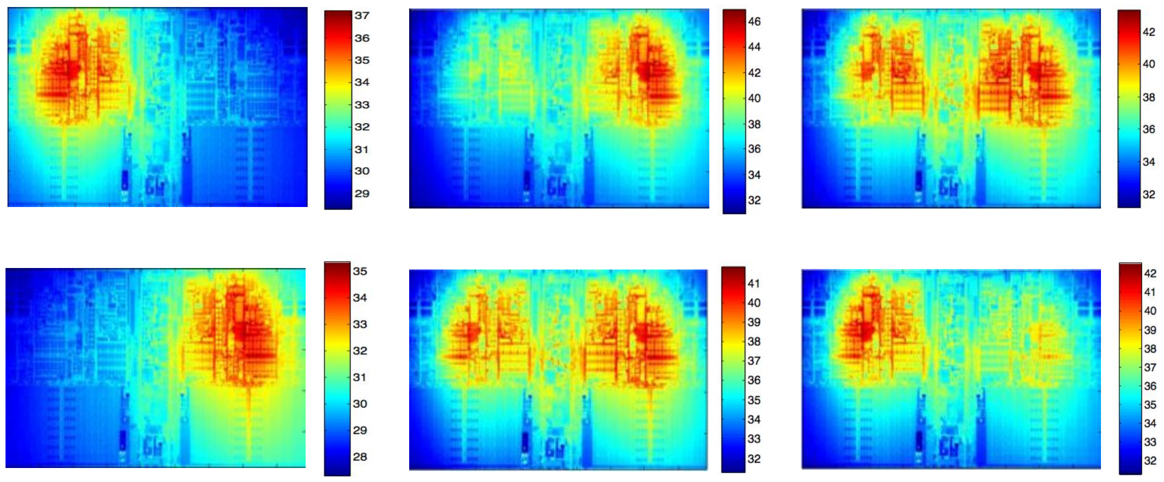
Figure 3.4: Examples of thermal traces of different applications on a dual-core AMD Athlon II.

The processor used for our experiments is a dual-core 45 nm AMD Athlon II X2 240 processor with die dimensions of 14 × 8.5 mm. To collect the thermal traces for use in the experiments, we use the CPU SPEC 2006 benchmark suite which includes 29 applications. The benchmarks have a number of integer and floating point workloads that cover a wide range of applications such as compilers, data compression algorithms, artificial intelligence algorithms, finite element analysis, and ray tracing. Figure 3.4 gives a number of thermal images captured from the execution of the SPEC CPU 2006 workloads. The thermal images demonstrate that within-die thermal gradients can reach up to 16 °C, and that differences in workloads can lead to strong variations in hot spot locations.

Comparison between Regular Sink and Oil Sink. It is imperative to contrast the thermal behavior of the processor using a regular metal heat sink with a fan and the proposed infrared-transparent oil heat sink. Previous works in the literature show that the heat sink setup has an impact on the thermal behavior [48, 80]. To contrast the thermal responses of these two different heat sinks, we rely on the measurements of two embedded thermal sensors in the processor; the infrared camera is irrelevant in this experiment. We monitor the embedded thermal sensors of the same processor using these two heat sink setups

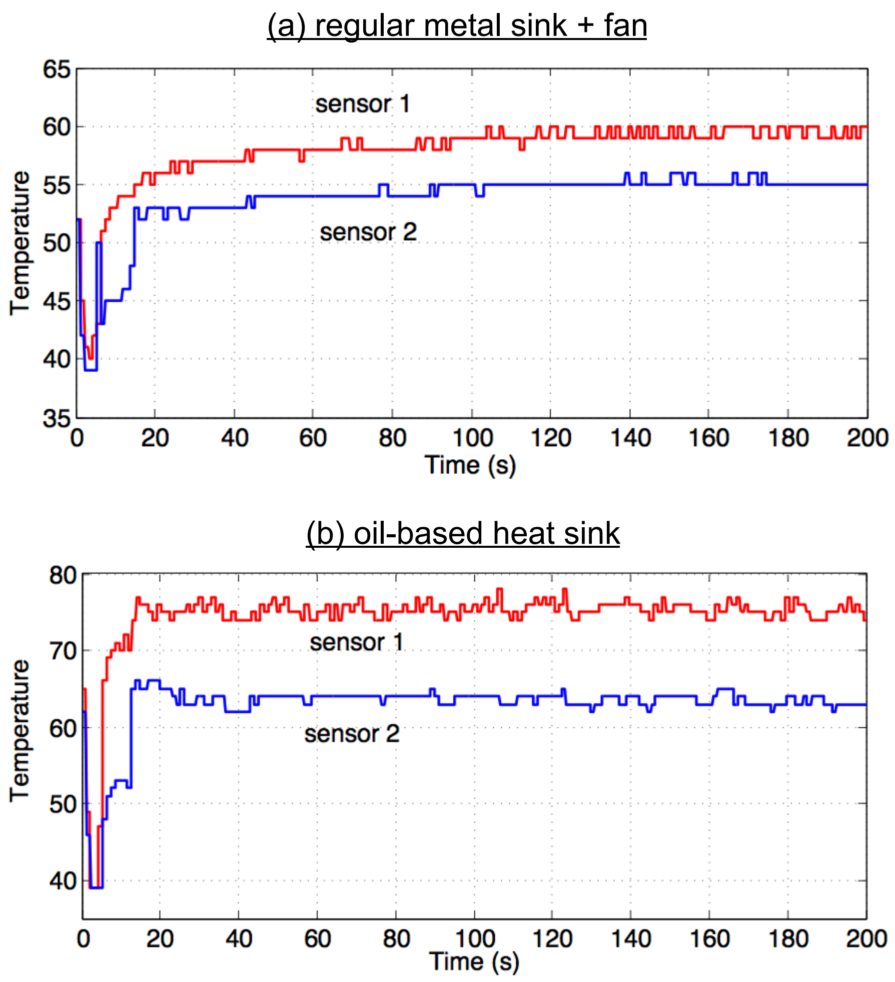

Figure 3.5: Thermal sensors measurements for the first 200 seconds of the `gamess` and `soplex` workloads running in parallel.

under identical workloads. Figure 3.5 gives the thermal sensors measurements for the first 200 seconds of the `gamess+soplex` workloads. Figure 3.5.a gives the measurements while the processor is coupled with the regular heat sink and fan setup, while Figure 3.5.b gives the measurements while the processor is coupled with the machined oil sink setup. The measurements in Figure 3.5 lead to the following three observations.

1. The measurements show that the two setups give spatially and temporally correlated results. That is, if sensor 1 gives a higher measurement than sensor 2 in our setup at some point in time, then it will also give higher temperature in the other setup at the

same time. This result is important because the hot spot is by definition the highest temperature on the die, and thus if we generalize the result of this experiment, we conclude that changing the heat sink setups does not alter the location of the hot spot. We have repeated Experiment 1 using different workloads, and we found that this result holds.

2. The measurements show that thermal gradients exists with the two setups. The metal sink setup shows a gradient of 5°C while the oil sink setup shows a gradient of about 10°C. This result suggests that perhaps a simple affine transformation can make the measurements from the oil sink look like those obtained from the regular metal heat sink.

3. The results show that the tested processor updates its thermal sensors every one second. This updating rate is controlled by the strobing rate of the ADC of the digital thermal sensors. High strobing rates consume a good amount of processor power during runtime. The embedded sensor update rate is far slower than our infrared imaging system which can update its measurements every 10 ms. In some experiments we observed quick, yet small variations in temperature detected by the infrared imaging system that were missed by the embedded sensors.

Characterization of Hot Spot Locations. We execute the 29 SPEC CPU workloads on the processor to collect tens of thousands of thermal traces during the runtime operation of the processor. We execute the workloads in single and dual workload configurations. Using a threshold of 37 °C, we identify the hot spot location in each trace and then we plot all the identified hot spot locations in Figure 3.6. The points in the figure give the set of potential locations where hot spots can occur during runtime. The hot spot locations are generally localized at and around the centers of the two cores and in the common memory controller area between the cores. These hot spot locations are plausible as designers most likely placed the frequently used core units towards the center to facilitate interconnections to other functional units. The L2 caches are consistently the coolest areas of the processor. The large range of possible hot spot locations for just two cores demonstrates the need for multiple sensors to accurately track hot spots during runtime.

As with any computer design experience, designers must choose a set of representative workloads to tune and benchmark their architectural choices. To confirm that the SPEC CPU 2006 workloads are representative enough for the purposes of thermal research, we re-perform the characterization with other benchmark workloads. We evaluate the hot spot locations for the SPEC power_ssj 2008 benchmark, which is a multi-threaded transactional workload that has a client-server model of operation, and we also create our own micro-benchmark which is a L1-cache resident floating-point kernel. We plot the locations of the hot spots of the power_ssj workload in Figure 3.7. The figure shows that these locations are a subset of the locations identified for SPEC CPU workloads and given in Figure 3.6. The hot spot locations of the created micro benchmark are given in Figure 3.8, and also form a subset of the locations in Figure 3.6. These results convincingly show that SPEC CPU 2006 is representative enough for thermal research.

3.2 Thermal Sensor Allocation Techniques

If thermal sensors did not incur an area overhead, sensor allocation would be trivial. In reality, they do incur a significant area overhead. A recent design study shows that a digital thermal sensor can occupy an area of 500 μm $\times$ 164 μm in 32 nm [119]. Thus it is necessary to find the best sensor placement given a budget area and number of sensors. Previous techniques used the k-means algorithm to allocate sensors at the centers of k hot spot clusters. A concern with this approach arises when the number of available sensors is less than the number of identified hot spots. In this case, the cluster centers could be relatively far from the hot spot locations, and thus inaccurate thermal tracking can occur.

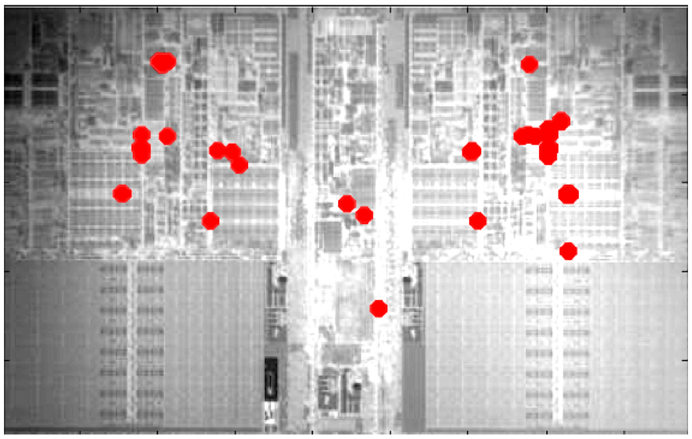

Figure 3.6: Locations of hot spots of SPEC CPU workloads on AMD Athlon II processor.

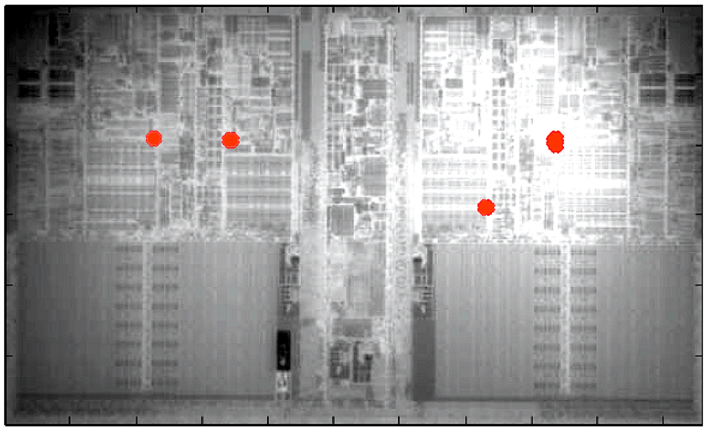

Figure 3.7: Locations of hot spots of SPEC powerssj transactional workload on AMD Athlon II processor.

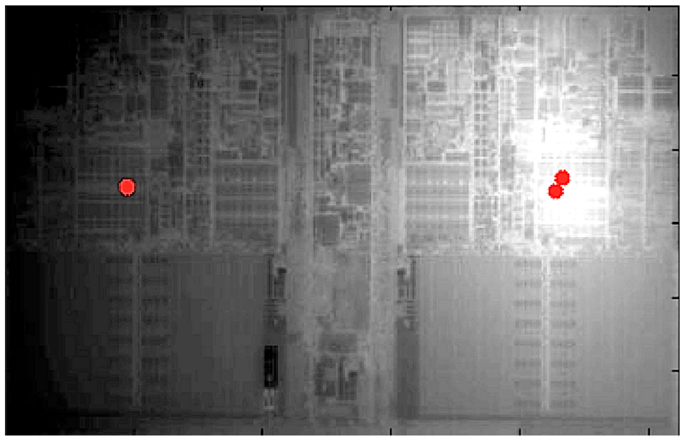

Figure 3.8: Locations of hot spots of created microbenchmark on AMD Athlon II processor.

Given a discretized set of p die locations, we define a *thermal trace* $\mathbf{T}_i$ as a vector of length p that gives the temperature at each die location in trace i. If S denotes the set of locations of the thermal sensors, we will use the notation $\mathbf{T}_i(S)$ to denote the thermal measurements at the locations specified by S for trace i. We define the *tracking error*, e_i, of thermal trace i as the difference between the maximum temperature (or hot spot) of the trace and the maximum of the measurements of thermal sensors, i.e., $e_i = \sup(\mathbf{T}_i) - \sup(\mathbf{T}_i(S))$. We define the *sensor error* as the worst tracking error across all traces. We believe that minimizing the worst error is more relevant than the average error because DTM methods are designed to take care of thermal emergencies triggered by extreme temperatures. We define the hot spot tracking problem as follows.

Hot spot tracking problem: Given a budget of k sensors and n thermal characterization traces $\mathbf{T}_1, \ldots, \mathbf{T}_n$, find the set of sensor locations S that minimizes the sensor error across all n traces i.e., $\min \sup\{e_1, \ldots, e_n\}$, and such that $|S| = k$.

Theorem 1. The hot spot tracking problem is NP-hard.

Proof: We reduce the classical NP-hard vertex cover problem to the hot spot tracking problem. Consequently, if a polynomial time algorithm exists for our problem then P $=$ NP. Given a graph that consists of a set of vertices V and a set of edges E, the objective of the k-cover problem is to decide if there exists k vertices that are incident on all edges [36]. Our polynomial-time reduction transforms a graph instance to a hot spot tracking instance as follows. If the graph has $|V| = p$ vertices, each labeled with a number between 1 and p, then for every edge $d_j = \{u, v\}$, we generate a trace vector $\mathbf{T}_j$ of length $|V|$ where the trace is zero everywhere except at locations u and v. That is, $\mathbf{T}_j(\{u, v\}) = 1$ and 0 elsewhere. By construction, $\forall j : \sup(\mathbf{T}_j) = 1$ and the location of a sensor corresponds to a vertex label. If a polynomial-time algorithm for the hot spot tracking problem exists, then given the thermal traces T_j's and k, the algorithm will return a sensor error of 1 if it cannot find k locations that track all hot spots (and hence no k-vertex cover), and it will return a sensor error of 0 if it finds k locations (and hence a k-vertex cover).

With NP-hardness established, an optimal algorithm (e.g., integer linear programming or branch and bound) to the problem would require exponential runtime as a function of the number of discrete locations on the die and the number of sensors. Thus, we propose a heuristic solution that consists of two phases. The *first phase* is constructive in nature and produces an initial set of locations, and the *second phase* is iterative in nature and improves upon the results of the first phase. We describe these two phases in the next paragraphs.

Constructive Phase: The objective of the first phase is to find a good initial set of sensor locations. We think a good initial allocation should capture the hot spots at a number of *independent* locations, as dependent locations will lead to sensor placements that are extremely close and clustered around one or two hot spot locations. Before describing the proposed algorithm, we introduce some notation. We first use the given thermal traces $\mathbf{T}_1, \ldots, \mathbf{T}_n$ to construct a $n \times p$ matrix *characterization matrix* $\mathbf{C}$ where each row consists of the measurements of a temperature trace vector. We will use the notation $\mathbf{C}_S$ to denote the matrix formed from the set of columns of $\mathbf{C}$ with indices in the set S.

To achieve our objective, we propose an algorithm in Figure 3.9 that is inspired by matrix volume sampling techniques [16]. The algorithm maintains two sets S and L: S is the set of chosen sensor locations and L is the set of available sensor locations. In Step 1, S is empty and L is initialized with all possible p locations. In Step 2, the algorithm picks the location with the highest temperature based on the L_2 norm and accordingly updates S

Procedure: Initialize locations of sensors
Input: Characterization data $\mathbf{C}$ as a $n \times p$ matrix
Output: Locations of k thermal sensors

```
Let $S = \emptyset$ and $L = \{1, \ldots, p\}$
Let $s_1 = \arg\max_{s \in L} ||\mathbf{C}_{\{s\}}||_2$
Let $S = S \cup \{s_1\}$ and $L = L - \{s_1\}$
For $i = 2, \ldots, k$:
  Project $\mathbf{C_L}$ into the column space of $\mathbf{C}_S$:    $\mathbf{P} = \mathbf{C}_S \mathbf{C}_S^{\dagger} \mathbf{C}_L$
  Find the orthogonal components: $\mathbf{N} = \mathbf{C}_L - \mathbf{P}$
  Let $s_i = \arg\max_{s \in L} ||\mathbf{N}_{\{s\}}||_2$
  Let $S = S \cup \{s_i\}$ and $L = L - \{s_i\}$
Return $S$
```

Figure 3.9: Construction procedure to initialize locations of thermal sensors.

and L in Step 3. The algorithm then iteratively (Steps 5–8) computes the orthogonal components of the column vectors at the available sensors locations (Steps 5 and 6), and then in Step 7, it picks the location with highest orthogonal L_2 norm. The projection in Step 7 is carried out using standard linear algebra techniques where $\mathbf{C}_S^{\dagger}$ is the Moore-Penrose inverse which is defined as $\mathbf{C}_S^{\dagger} = (\mathbf{C}_S' \mathbf{C}_S)^{-1} \mathbf{C}_S'$ (where $\mathbf{C}_S'$ is the transpose of the matrix $\mathbf{C}_S$) [107]. The iterations are repeated until k locations are determined.

Iterative Phase: The iterative procedure takes as inputs the locations of the sensors as computed from the constructive phase and the characterization matrix, and then it iteratively adjusts the sensor locations. The algorithm given in Figure 3.10 selects one sensor at a time, while locking the locations of the other sensors, and then finds (by enumerat-

Procedure: Iterative improvement procedure
Input: Characterization data **C** and initial placement S
Output: Locations of k thermal sensors

```
Let $e_p = \infty$
Do:
  For $i = 1 \ldots k$
    Let $S' = S - \{s_i\}$
    Find location $s_j$ such that $S' = S' \cup \{s_j\}$ gives least sensor error
    Let $S = S' \cup \{s_j\}$
While $\frac{e_p - e}{e_p} < 0.001$
Return $S$
```

Figure 3.10: Iterative procedure to improve the locations of thermal sensors.

ing at the p possible locations) the best new location for the selected sensor to minimize the sensor error (Step 5). The sensor is inserted into the best new location (Step 6), and the process is repeated for the next sensor until all sensors are selected. This constitutes one iteration, which is repeated until the improvement in sensor error drops below a predetermined threshold as given by Step 7 (0.1% in our implementation).

Experimental Results. The inputs to our sensor allocation algorithm are the locations of the hot spots and temperatures at a discrete set of die locations. These inputs could come from either computer-based thermal simulations and/or real data measured from infrared imaging equipment. The majority of previous work on algorithmic techniques for thermal tracking relied on computer-based simulations. In these simulations power traces from architectural simulators (e.g., Wattch[13]) are fed together with a processor's block-level layout to a thermal simulator (e.g., Hotspot [49]) to obtain thermal traces. We evaluate our sensor allocation algorithm using high-resolution thermal traces obtained from direct thermal imaging as discussed in Section 3.1. Recent studies on power and temperature modeling confirm the value of the complementary information that thermal imaging provides [81, 37, 48].

We implement three allocation techniques: `uniform` where the sensors are uniformly spaced as a grid, `k-means` which is an implementation of the thermal-aware k-means algorithm developed by Memik *et al.* [78], and the `proposed` thermal sensor allocation algorithm. We report the sensor error for $k = 1$ to 6 in Figure 3.11, from which we observe the following:

- The `uniform` technique can lead to large sensor errors (10°C), and the error does not decrease monotonically as the number of sensors is increased. This result is plausible as an allocation with fewer sensors could land by "luck" a sensor nearby a

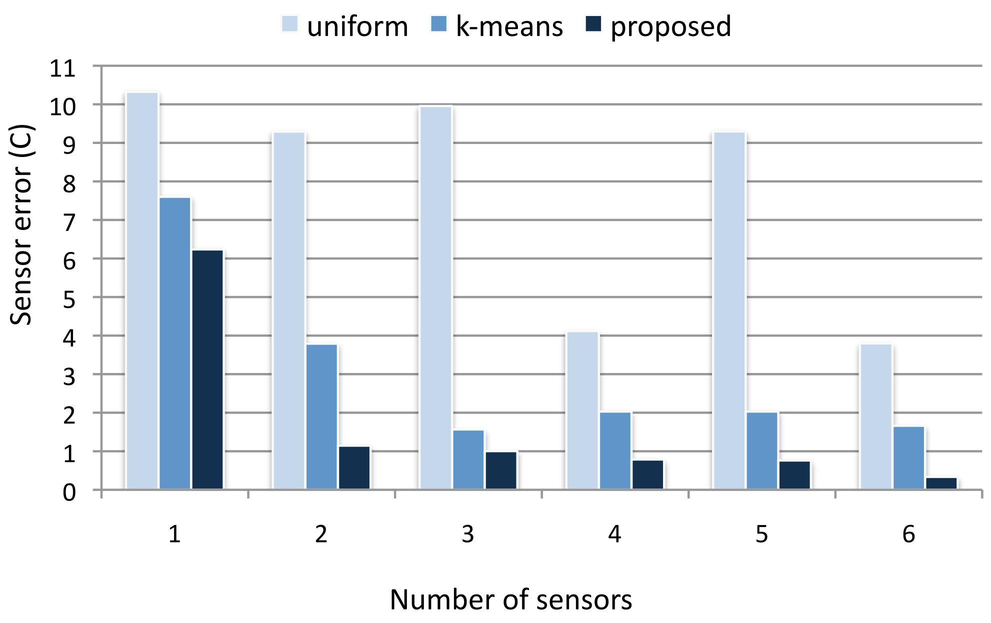

Figure 3.11: Average thermal sensor error as a function of allocation technique and number of sensors for AMD processor.

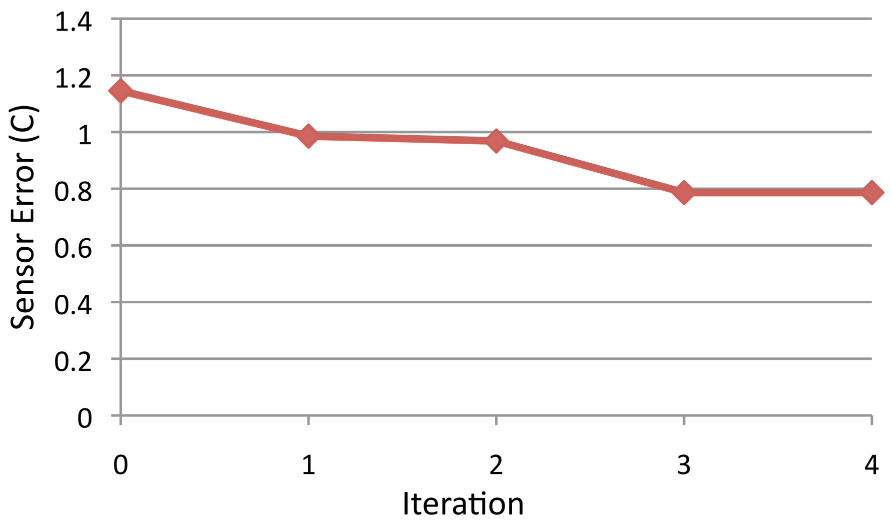

Figure 3.12: Sensor error for $k = 4$ as a function of the number of iterations.

hot spot reducing the error. Uniformly allocated sensors should be avoided.

- The `k-means` algorithm produces better results than uniform allocation but it is outperformed by the `proposed` algorithm. For $k = 2$, `k-means` gives 3.79°C error versus the proposed method which gives 1.15°C error. One of the problems of `k-means` is that it does not take into account the frequency of hot spots at a particular location in its clustering criteria, and it does not directly optimize for the sensor error but rather a combined metric of location proximity and the average temperature error.

To provide a better understanding of the contribution of the constructive and iterative phases towards the final solution, we plot in Figure 3.12 the sensor error for $k = 4$ as a function of the number of iterations. Iteration 0 is the result of the initial constructive phase. The plot shows that it takes about 4 iterations until the improvement in sensor error in the last iteration is less than 0.1% from the prior iteration, and that the initial construction phase by itself produces better results than the k-means approach.

sensors	k-means		proposed	
	avg (mm)	max (mm)	avg (mm)	max (mm)
1	3.32	4.76	3.10	5.08
2	0.99	4.63	0.93	5.38
3	0.85	5.14	0.57	4.92
4	1.18	5.43	0.42	4.86
5	1.30	5.43	0.44	5.43
6	1.07	4.42	0.31	3.69

Table 3.1: Average and maximum distance (in mm) between location of sensor reporting highest temperature and true hot spot location.

In addition to temperature tracking, it is necessary to identify the locations of hot spots. Identifying the hot spot locations enables the DTM system to take appropriate actions in preventing a thermal emergency. For example, the DTM system can reduce the frequency of an overheating functional unit or migrate workloads away from it. To illustrate the advantage of our proposed algorithm in comparison to previous methods, we compute for each thermal trace the distance between the location of the hot spot and the location of the sensor reporting the highest temperature. We then compute the average and maximum distances across all thermal traces. We report our results in mm in Table 3.1. The results show that our proposed algorithm is always closer to the true hot spot location on average across all traces. This proximity enables the DTM system to make better informed decisions to prevent thermal emergencies at the correct functional units.

3.3 Soft Sensing Techniques

In this section we propose the concept of *soft sensing* to augment the embedded hard sensors and improve thermal tracking. We think that there are a number of possible scenarios that would benefit from soft sensing:

- Because of design constraints, designers might not be able to insert hard sensors in the desired locations. As a result, the measurements of the hard sensors will no longer accurately track the temperatures of the hot spots. Soft sensors will allow designers to overcome this problem by intelligently combining hard sensor measurements to estimate the temperatures at the most problematic locations.

- Digital thermal sensors consume die area and thus designers limit their numbers at the expense of thermal tracking accuracy. Soft sensing enables designers and runtime thermal management systems to circumvent the limitations on the number of sensors and potentially achieve *full thermal characterization*, where temperatures are estimated at all possible locations.

We propose two techniques for soft thermal sensing. In Subsection 3.3.1 we propose soft sensing techniques that are enabled by the availability of *a priori* thermal characterization data from either simulations or infrared imaging. In Subsection 3.3.2 we develop a general-purpose technique for soft sensing that does not require any *a priori* characterization.

3.3.1 Proposed Soft Sensing with A Priori Characterization

In our proposed technique, a soft sensor measurement is equal to a weighted linear combinations of the measurements of the hard sensors. Thus, if $\hat{T}_i(l)$ denotes the estimated soft sensor temperature at location l at time instance i, then this measurement can be expressed by

$$\hat{T}_i(l) = \sum_{j=1}^{k} w(l,j) T_i(s_j), \tag{3.1}$$

where $T_i(s_j)$ is the measurement reported in trace i by the sensor placed at location s_j, and $w(l, j)$ is a weight that is a function of locations l and j. To determine the best set of weights for a location l, we utilize available thermal characterization traces to learn the optimal linear combination of hard sensor measurements. Given n traces, we can construct the following set of equations:

$$\begin{aligned}
\mathbf{A}\mathbf{w}_l &= \begin{pmatrix} T_1(s_1) & \cdots & T_1(s_k) \\ \vdots & \vdots & \vdots \\ T_n(s_1) & \cdots & T_n(s_k) \end{pmatrix} \begin{pmatrix} w(l,1) \\ \vdots \\ w(l,k) \end{pmatrix} \\
&= \begin{pmatrix} T_1(l) \\ \vdots \\ T_n(l) \end{pmatrix} = \mathbf{b}_l, \qquad (3.2)
\end{aligned}$$

which can be written succinctly in matrix notation as $\mathbf{A}\mathbf{w}_l = \mathbf{b}_l$. The best set of weights that minimizes the total least square error can be computed $\mathbf{w}_l = (\mathbf{A}'\mathbf{A})^{-1}\mathbf{A}'\mathbf{b}_l = \mathbf{A}^{\dagger}\mathbf{b}_l$.

Computing the optimal weights can be carried only once off-line either (1) during design time using the results from thermal modeling and simulation tools, or (2) after fabrication when infrared imaging techniques are typically used to characterize and calibrate the embedded thermal sensors. The computed weights can be stored in configurable registers of the processors. During runtime, these weights together with the hard sensor measurements are retrieved and used by the DTM system to improve thermal tracking. For example, if three hard sensors are deployed then only three multiplications and two additions are required for every additional location where the temperature needs to be estimated. In our experiments, we found that most processors update their internal thermal sensors every second. At this rate, the computational overhead is minimal.

Experimental Results. To model the impact of design constraints, we design an experiment in which the sensors are embedded *near* the locations identified by the sensor allocation algorithm rather than at the exact locations. We assume the thermal setup of Section 3.1 and the sensor locations identified by the algorithm proposed in Section 3.2. We assume that the locations identified by the sensor allocation algorithm are infeasible due to design constraints, and we perturb these locations by a random distance that is drawn from a Gaussian distribution with a standard deviation of 1 mm. For example, Figure 3.13 shows the original locations of the sensors (for $k = 3$) and the new locations after perturbation. Using the measurements of the perturbed hard sensors, the soft sensing technique is used to estimate the temperatures at the possible hot spot locations. We then compare these estimates to the measurements reported from the infrared imaging system. Figure 3.14 gives the error between the hot spots and thermal sensors using the hard sensors alone and the hard + soft sensor technique. Compared to the earlier results of Figure 3.11, it is clear that the small perturbations in the hard sensor locations could introduce unpredictable errors in their measurements. The results also show that the soft sensing technique is capable of reducing the tracking error. For example, when $k = 3$, soft sensing cuts down the tracking error from 9.35°C to 4.44°C.

Since embedded hard sensors could exhibit noise in their measurements [56, 122], we assess the impact of this noise on soft sensor measurements. We superimpose independent normally distributed noise for each hard sensor, and then compute the tracking error of the soft sensors. In Table 3.2 we report the error as a function of the number of sensors and the standard deviation of the noise distribution. The second column gives the error if no noise is assumed for the hard sensors (same results as in Figure 3.14) and the subsequent columns show the expected trend of increasing soft sensor error as the noise increases in the hard sensor measurements.

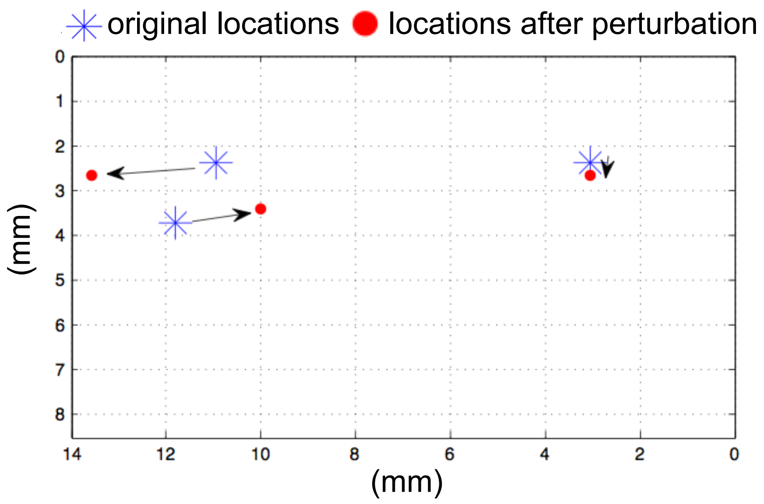

Figure 3.13: Location of sensors before and after perturbation.

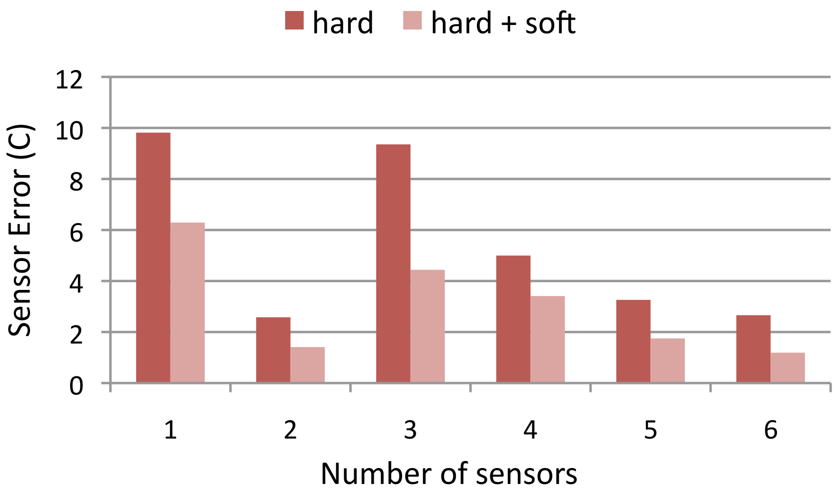

Figure 3.14: Impact of soft sensing after sensor perturbation.

3.3.2 Proposed Soft Sensing Using Spectral Techniques

In this section, we develop soft sensing techniques for scenarios in which the only available information comes from a set of sensors at known die locations. Without detailed *a priori* thermal characterization, the relationship between the sensor measurements and soft sensor locations must be inferred using assumptions about the nature of the tempera-

sensors	standard deviation					
	0°C	0.2°C	0.4°C	0.6°C	0.8°C	0.1°C
1	6.29	6.39	6.55	6.76	6.77	6.91
2	1.41	1.49	1.60	1.79	2.01	2.12
3	4.44	3.58	3.72	3.79	4.00	4.14
4	3.41	3.52	3.21	3.35	3.27	3.50
5	1.75	1.89	2.15	2.40	2.60	2.80
6	1.19	1.23	1.29	1.66	1.75	1.78

Table 3.2: Soft sensor error as a function of the error in hard sensors.

ture signal. We know that the spatial temperature profile is a low-pass filtered version of the power density profile and can be represented sparsely in the frequency domain. Thus, we develop frequency-domain signal reconstruction techniques in order to interpolate unknown temperatures as a linear combination of the known sensor values.

Our approach is formally grounded in spectral Fourier analysis techniques. Application of these techniques to temperature sensing is based on recognition that die temperature is simply a space-varying signal, and that space-varying signals are treated identically to time-varying signals in Fourier signal analysis. While temperature is a continuous variable, any representation in computer memory must be discretized. Thus, the temperature $t(m,n)$ is a discrete function that is defined over a finite region $0 \leq m \leq M-1$ and $0 \leq n \leq N-1$, where M and N are the *resolutions* required for thermal characterization. For example, if a die has dimensions $1\ cm \times 1\ cm$, then with resolutions $M = N = 128$, temperatures are evaluated for every $78\ \mu m \times 78\ \mu m$ square. The two-dimensional Discrete Fourier Transform (DFT) is given by

$$T(p,q) \quad = \quad \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} t(m,n) e^{-j2\pi pm/M} e^{-j2\pi qn/N} \tag{3.3}$$

for all $p = 0, 1, \ldots M - 1$ and $q = 0, 1, \ldots N - 1$, and the inverse DFT is given by

$$t(m, n) \quad = \quad \frac{1}{MN} \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} T(p, q) e^{j2\pi pm/M} e^{j2\pi qn/N} \tag{3.4}$$

for all $m = 0, 1, \ldots M - 1$ and $n = 0, 1, \ldots N - 1$. To compute the DFT and the inverse DFT efficiently, the Fast Fourier Transform (FFT) and the inverse FFT are used, achieving an $O(MN \log MN)$ runtime as opposed to $O((MN)^2)$ [91].

Impact of Sampling on Frequency-Domain Representation of Temperature. Consider the case in which the locations of the sensors are aligned on a lattice such that they are equally spaced from each other. There are a number of lattices for which such uniformity can be achieved, including hexagonal lattices, diamond lattices, and rectangular grids. We focus on rectangular grids as they are more suitable for the tiled layouts typically encountered in multi-core processors and GPUs. To sample the temperature $t(m, n)$, we assume that the thermal sensors have been placed with a horizontal/vertical spacing of $P \in \mathcal{Z}^+$. Thus, the sampling signal $s(m, n)$ can be described with

$$s(m, n) = \sum_{u=0}^{\lfloor \frac{M-1}{P} \rfloor} \sum_{v=0}^{\lfloor \frac{N-1}{P} \rfloor} \delta(m - uP, n - vP), \tag{3.5}$$

where $\delta(\cdot, \cdot)$ is the Dirac delta function. To obtain the sampled temperature signal $t_s(m, n)$, we multiply the temperature with the sampling function to get

$$\begin{aligned} t_s(m,n) &= t(m,n)s(m,n) \\ &= t(m,n)\sum_{u=0}^{\lfloor\frac{M-1}{P}\rfloor}\sum_{v=0}^{\lfloor\frac{N-1}{P}\rfloor}\delta(m-uP,n-vP) \\ &= \sum_{u=0}^{\lfloor\frac{M-1}{P}\rfloor}\sum_{v=0}^{\lfloor\frac{N-1}{P}\rfloor} t(uP,vP)\delta(m-uP,n-vP) \end{aligned} \tag{3.6}$$

Consider the DFT of $t_s(m,n)$. Since $t_s(m,n)$ is the product of $t(m,n)$ and $s(m,n)$, then the DFT $T_s(p,q)$ of the sampled signal $t_s(m,n)$ is equal to the convolution of the $T(p,q)$ and $S(p,q)$, which are the DFTs of $t(m,n)$ and $s(m,n)$ respectively. The Fourier transform of a periodic impulse train is a periodic impulse train as well; i.e.,

$$\sum_{u=0}^{\lfloor\frac{M-1}{P}\rfloor}\sum_{v=0}^{\lfloor\frac{N-1}{P}\rfloor}\delta(m-uP,n-vP) \leftrightarrow \frac{1}{P^2}\sum_{u=0}^{P-1}\sum_{v=0}^{P-1}\delta(p-u\frac{M}{P},q-v\frac{N}{P}) \tag{3.7}$$

Thus, the convolution of $T(p,q)$ and $S(p,q)$ gives

$$T_s(p,q) = \frac{1}{P^2}\sum_{u=0}^{P-1}\sum_{v=0}^{P-1}T(p-u\frac{M}{P},q-v\frac{N}{P}). \tag{3.8}$$

Temperature sampling by the thermal sensors in the space-domain has led to periodic copies of the Fourier transform of the temperature signal in the spectral domain. Figure 3.15 visually illustrates the impact of sampling, where Figure 3.15.a gives a thermal map of a 16-core processor in both the space and spectral-domain. After sampling, the spectral-domain representation of Figure 3.15.b shows the repetition of the spectral map of Figure 3.15.a. According to the Nyquist-Shannon theorem, if copies of the spectral-domain temperature signals are spread far "enough" apart, then overlap or *aliasing* between the copies

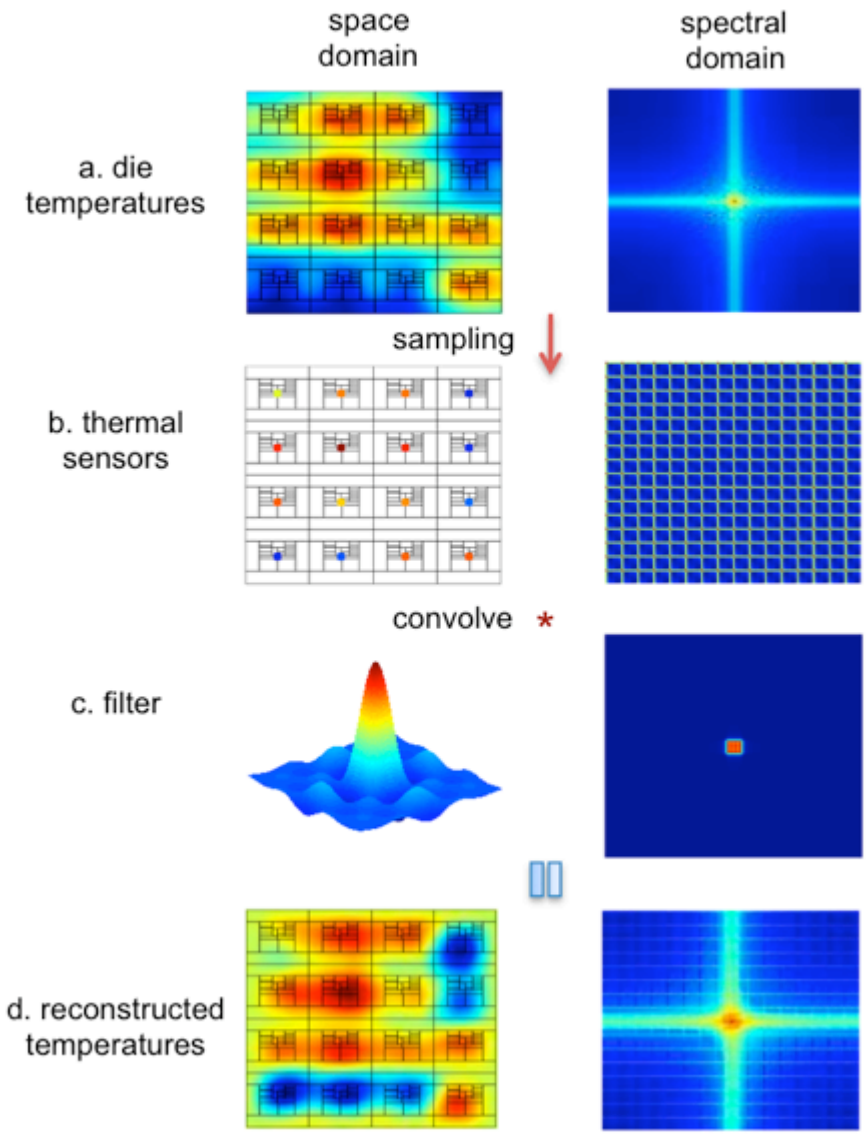

Figure 3.15: Main steps used for signal reconstruction. The log of the magnitude of the 2D DFT is plotted.

will not occur. The necessary spreading, which is controlled by the sampling frequency, depends on the highest frequency seen in the temperature signal. If the temperature signal is band-limited with frequency B, then sampling at a rate higher than $2B$ guarantees full reconstruction of the original signal. Sampling at a higher rate is achieved by decreasing the spacing P between the thermal sensors. Thus, thermal sensors should be spaced such that $1/P \geq 2B$, or equivalently $P \leq 1/(2B)$. To minimize information loss due to sampling, one must pick a bandwidth B below which most of the signal's energy is concentrated.

A classical result in signal processing states that time-limited signals are not band-limited, and that band-limited signals are not time-limited; i.e., a signal cannot be simultaneously time-limited and band-limited [96]. Temperature, a space-varying signal, is space-limited by the edges of the chip. Thus, the spectral representation of the temperature is not band-limited. As a result, perfect reconstruction is not possible in the case of on-chip temperatures; however, near-perfect reconstruction with negligible loss of information is possible if these *edge effects* are appropriately handled during reconstruction such that the higher frequency magnitudes in the spectral domain are minimized. Techniques for handling edge effects are discussed next.

Thermal Reconstruction from Samples. We have seen that sampling a signal in the space-domain leads to periodic copies of the Fourier transform, $T(p, q)$, of the original signal $t(m, n)$, in the spectral-domain. If the temperature signal is band-limited (or has negligible energy beyond a certain frequency) then these periodic copies are well separated from each other and aliasing is minimal. The Whittaker-Shannon-Kotelnikov (WSK) classical theorem states that to recover the original signal from the samples, it is sufficient to extract only one copy of the signal in the spectral-domain [96]. This extraction can be achieved using a low-pass box filter as shown in Figure 3.15.c. In the frequency domain, this box can be expressed as

$$\begin{aligned} F(p,q) &= 1 \text{ if } \quad |p| \leq B \quad \text{and} \quad |q| \leq B \\ &= 0 \text{ otherwise.} \end{aligned} \tag{3.9}$$

Taking the inverse DFT of the box filter gives the spatial-domain representation of the filter $f(m,n)$ which is equal to

$$f(m,n) = \text{sinc}(\frac{m}{B})\text{sinc}(\frac{n}{B}). \tag{3.10}$$

Reconstruction is achieved by *convolving* the space-domain samples with the space-domain filter representation. That is, the reconstructed temperature of a chip $t_r(m,n)$ can be found using

$$t_r(m,n) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} t_s(u,v)\text{sinc}(\frac{m}{B} - u)\text{sinc}(\frac{n}{B} - v). \tag{3.11}$$

This result is illustrated in Figure 3.15.d. One of the practical problems that arises when using the sinc function is that it is not space-limited. In any implementation, the sinc function must be truncated, which has the effect of smearing its spectral-domain representation, leading to a less than sharp box filter edge. This smearing effect can be minimized by *windowing* the sinc function. In our implementations, we multiply the sinc function by a *Hamming window* of the same size. The severity of the edge effects depends on the size of the sinc function used. Edge effects can be minimized by extending the temperature data by a distance larger than half the size of the filter function. The values in the extended region can be copies of the edge values, periodic repetitions of the temperature signal, or even a mirror image of the temperature signal. It is also useful to investigate other filter functions that approximate a low-pass box filter in the spectral domain without windowing [87]. We investigate three other functions.

- *Nearest neighbor:* The simplest interpolation function is nearest neighbor, in which each location is given a temperature equal to the value measured by the sensor closest to it. This is achieved by convolving the sampled temperature signal with a rectangular function expressed as follows:

$$\begin{aligned} f(x,y) &= 1, \text{ for } x \in [-0.5, 0.5] \text{ and } y \in [-0.5, 0.5] \\ f(x,y) &= 0, \text{ otherwise} \end{aligned}$$

- *Linear function:* Linear interpolation amounts to convolving the temperature samples with a round cone function. This function corresponds to a modestly good low-pass filter in the spectral-domain. However, it attenuates frequencies near the cut-off frequency, resulting in smoothing of the thermal characterization results. It also passes a good amount of energy above the cut-off frequency. The linear function is expressed as follows:

$$\begin{aligned} f(x,y) &= g(x)g(y), \text{ where} \\ g(u) &= (1-u) \text{ for } u \in [0,1] \end{aligned}$$

- *Cubic B-spline function:* Cubic B-spline functions are reasonably good low-pass filters. They are positive in the whole interval from 0 to 2, so they smooth somewhat more than necessary below the cut-off frequency. These filters are symmetric, so they only need to be expressed on the interval $[0, 2]$. The cubic B-spline function we

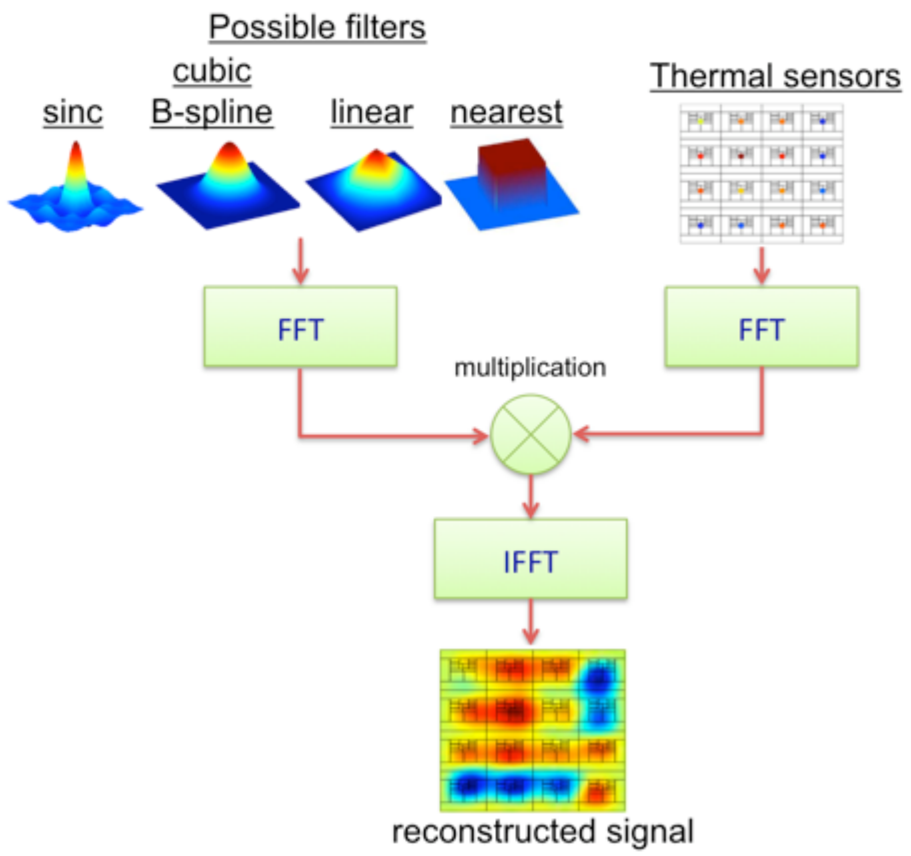

Figure 3.16: Flow of the proposed runtime thermal characterization technique.

use is expressed as follows:

$$
\begin{aligned}
f(x,y) &= g(x)g(y), \quad \text{where} \\
g(u) &= \frac{u^3}{2} - u^2 + \frac{4}{6} \quad \text{for} \ \ u \in [0,1] \\
g(u) &= \frac{-u^3}{6} + u^2 - 2u + \frac{8}{6} \quad \text{for} \ \ u \in [1,2]
\end{aligned}
$$

All three interpolation functions are contrasted with the sinc function in Figure 3.16. To achieve a computationally efficient thermal characterization, we propose the flow in Figure 3.16. Instead of directly convolving in the space-domain, we first take the Fast Fourier Transform (FFT) of the temperature samples and the space-domain representation of the interpolation functions. We then multiply the resultant spectral-domain representations to get the FFT of the reconstructed 2D thermal signal. We then apply the inverse FFT (IFFT) to get the full-resolution thermal characterization in the space-domain. If the required thermal resolution is $R = MN$, then application of the proposed flow of Figure 3.16 has a runtime of $O(R \log R)$, which gives a significant advantage over the $O(R^2)$ runtime achieved by straightforward convolution. Such speedup boost is necessary for runtime thermal characterization.

If thermal sensors are placed non-uniformly, then directly applying the proposed flow of Figure 3.16 could lead to large errors. Instead, it is necessary to apply signal reconstruction algorithms that are devised to handle non-uniform samples. One such algorithm is proposed by Sauer and Allebach [101]. This iterative algorithm consists of two steps:

1. Given the locations of the thermal sensors, construct a Voronoi diagram for the die. All die locations that belong to the Voronoi cell of a thermal sensor are assigned the same temperature as the thermal sensor at center of the cell.
2. The resultant 2D temperature map obtained from the first step is low-pass filtered B (by convolving the temperature with the sinc function) using the flow of Figure 3.16.

Sauer and Allebach [101] prove that iterating these two steps leads to convergence and reconstruction of the original signal if it is band-limited.

Experimental Results. To evaluate the effectiveness of our methodology, we set up a tool chain that simulates temperatures for a 32 nm 16-core processor. Our tool chain takes as inputs the processor's floor-plan and the workload that will run on each core and produces as output the steady-state temperatures at various grid locations. Using workload instruction traces, dynamic power traces for each micro-architectural unit are calculated and then fed together with the floor-plan into the thermal simulator. Once the steady-state temperatures are calculated, they are fed into a leakage calculator which outputs the corresponding leakage power of each unit. The leakage power values are then added to the original dynamic power values, and a new thermal simulation is performed. This process is iterated until the temperatures converge to stable values. Power and thermal simulations are performed using the following tools.

- For dynamic power estimation, we use a Wattch-like [13] power simulator, with the power consumption appropriately scaled to 32 nm technology based on ITRS predictions [53]. For the leakage consumption of the processor core units, we construct a leakage model using the expressions for leakage power from PTScalar [73]. To accurately model cache leakage power, we use CACTI 5.0 [113], which has accurate cache leakage values at current and future technology nodes.
- We utilize HotSpot (version 4.0) [47] for thermal simulation. HotSpot takes as inputs the processor floor-plan and workload power traces and produces as output the steady-state temperatures for a set of grid locations.
- We use the Alpha 21264 processor as our baseline core [59]. The 21264 is an out-of-order speculative execution core that is commonly used as a test-bench core in thermal management research [73, 75]. We create a 16-core processor based on the Alpha processor. The die area of the processor is $1.1\ cm \times 1.1\ cm$, and we discretize the temperature by defining a grid with resolution $M = 64 \times N = 64$, where each grid location represents the temperature for an area of size $172\ \mu m \times 172\ \mu m$.
- For workloads, we use eight benchmarks from the SPEC2000 suite [42]. We use four integer benchmarks: gcc, bzip, mcf, and twolf, and four floating point benchmarks: ammp, equake, lucas, and mesa.

In each simulation, we assign each core in the 16-core processor a workload from the SPEC2000 benchmark selection such that each core gets a random workload from the available eight. We then assign each core a random frequency in the range $1.5 - 3$ GHz. We then execute the tool chain to find the true temperatures at all grid locations. Thermal sampling is accomplished by zeroing temperatures at all grid locations except those corresponding to thermal sensors. The number of samples and the sample locations are varied, and each proposed method is evaluated for each sensor configuration using the *average* and *maximum* absolute post-reconstruction error across a set of 32 simulations.

In our first set of experiments, we determine the average and maximum absolute error for each reconstruction method for uniformly spaced samples while varying the total number of sensors on a 16-core processor. We present results for the following cases: 1 sensor per core, 4 sensors per core arranged on a 2×2 grid, and 9 sensors per core arranged on a 3×3 grid. The first case corresponds to 16 total sensors, while the second and third cases correspond to 64 and 144 total sensors respectively. The bar-plot of Figure 3.17 summarizes the average absolute error calculated for each proposed reconstruction method. We include error values for a geostatistical-based Kriging estimator for sake of comparison. The results show that as the number of sensors increases, the thermal characterization error decreases. Furthermore, the results show that our proposed reconstruction methods are capable of achieving full thermal characterization with minimum average absolute error ranging from 1.8% to 0.6%, depending on the number of the sensors. In this experimental setting, our convolution filters deliver near identical performance, and they significantly outperform the Kriging estimator, especially when the number of sensors are few. Our second set of experiments will reveal a differentiation in the performance of our proposed convolution filters. Using a single representative workload and frequency assignment, Figure 3.18 compares the thermal resolution attained by using 1, 4, and 9 sensors per core to the true thermal characterization. We include the latter case, despite a possibly unrealistic

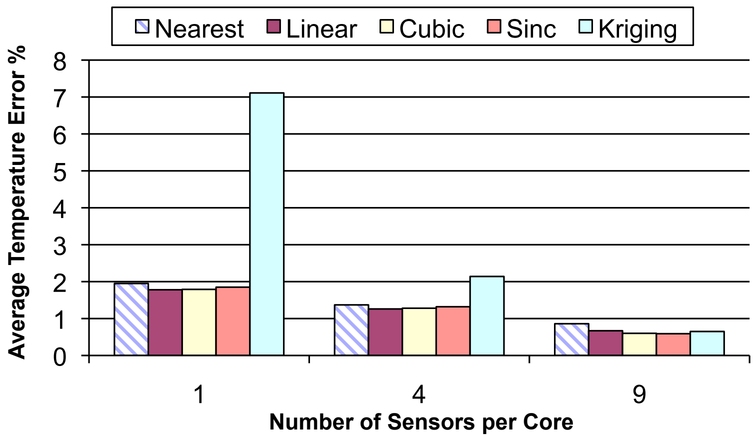

Figure 3.17: Results of full thermal characterization. We report the average error percentage for temperature estimation at all grid locations.

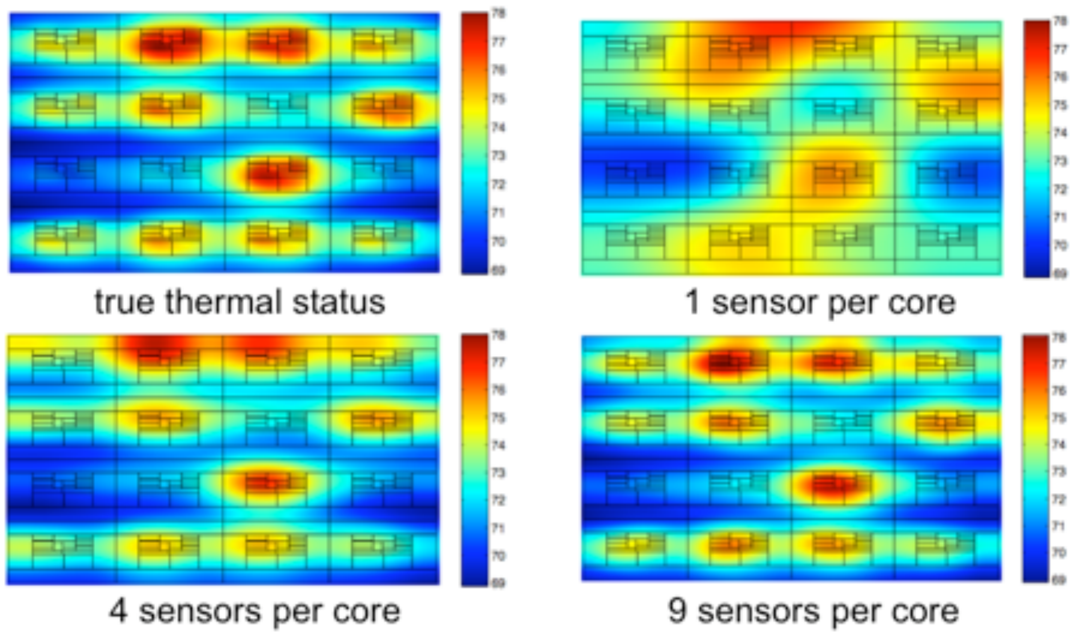

Figure 3.18: Impact of increasing the number of thermal sensors on the full thermal characterization.

number of total sensors for a 16-core processor, to illustrate that increasing the number of sensors eventually reconstructs the original signal to near perfection.

Full thermal characterization is particularly useful for advanced thermal management techniques, examples of which include workload scheduling and per-core frequency and

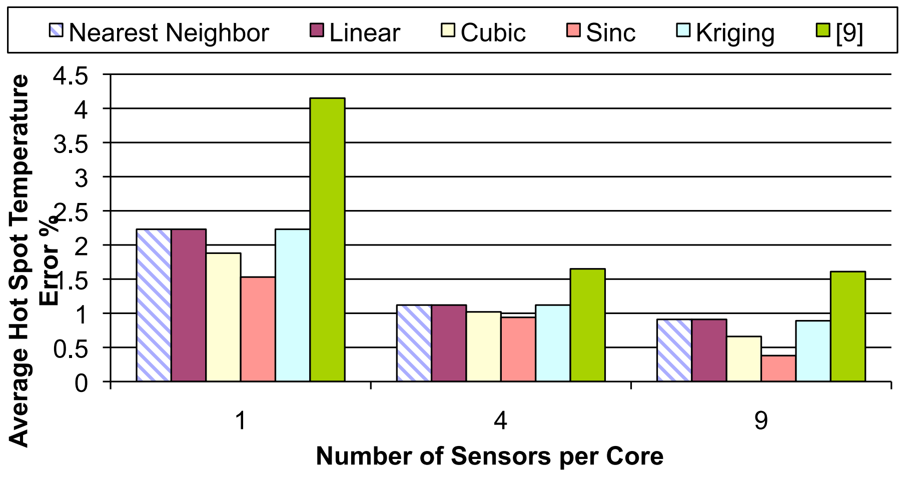

Figure 3.19: Hot spot estimation. We report the average error percentage for temperature estimation at the hottest die location.

voltage assignment. In simpler thermal management techniques (e.g., fan speed control), only the magnitude and the location of the maximum hot spot are relevant. Thus, in our second set of experiments, we consider the hypothetical performance of our methods in such applications. We first identify the location and magnitude of the maximum hot spot in the true thermal characterization. We then evaluate the temperature at that location in our reconstructed results and report the average absolute error. We report error values for each method explored in our first set of experiments, and in addition we implement and compare to the neighborhood interpolation scheme in [75]. Our results in Figure 3.19 show that the sinc filter function is consistently better at interpolating the maximum hot spot (0.4% errors for the case of 9 sensors per core), and that all of our reconstruction techniques outperform the Kriging estimator and the method proposed in [75].

In our third set of experiments, we consider non-uniform sensor placements. Figure 3.20 shows a representative workload and marks the locations of the thermal sensors in the processor's floor-plan with '*'. We use the same workload and frequency assignment as in Figure 3.18. Figure 3.20.a gives the output of the first step of the iterative algorithm which constructs the Voronoi diagram, and 3.20.b gives the results after the algorithm converges

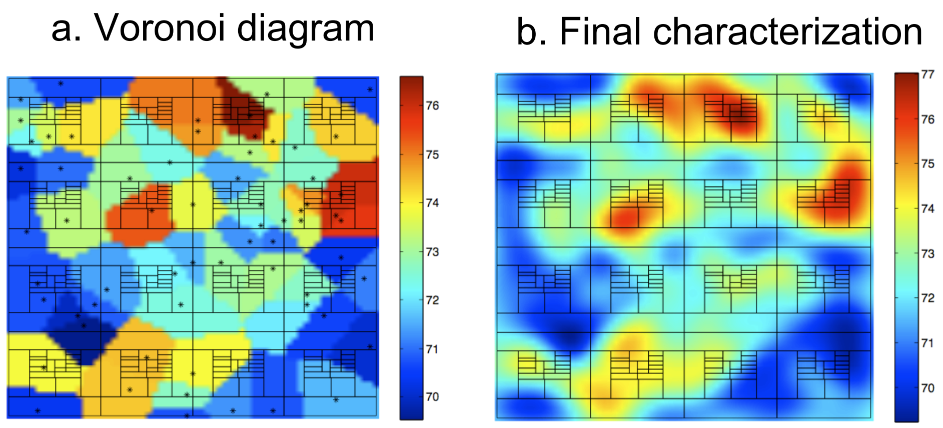

Figure 3.20: Results of thermal characterization using non-unform sampling.

using a sinc filter function. The average absolute error for full thermal characterization using non-uniform samples is found to be 2.39%. This value confirms that our methods are capable of handling signal reconstruction for both uniform and non-uniform sensor placements.

3.4 Summary

In this chapter we proposed new theoretical and experimental techniques to improve thermal sensing and hot spot thermal tracking in modern processors. We developed infrared imaging techniques to characterize the thermal behavior of real processors during runtime under different workloads. Our characterization demonstrate the extent of hot spots and thermal gradients in modern processors. Previous thermal characterization methods methods relied on simulations to generate the thermal characterization results; our experimental approach complements simulation-based techniques and provides a new perspective on the thermal behavior of real processors.

Based on the characterization results, we proposed a formulation for thermal sensor allocation to minimize the thermal tracking error during runtime. We proved that our formulation leads to a NP-hard problem and accordingly we proposed a heuristic solution method that is composed of constructive and iterative phases. The experimental results show that our method significantly improves upon methods in the literature.

Design constraints could force designers to use very limited number of sensors or to insert the sensors into non-ideal locations. To circumvent these limitations, we proposed two soft sensing techniques that combine the measurements of hard sensors to estimate the temperatures at the ideal locations. One technique leverages *a priori* design-time characterization data to seek customized weighted combinations to estimate the temperatures at any desired locations. The second technique does not require any *a priori* characterization data, and it leverages frequency-domain analysis techniques to estimate the temperatures from the measurements of the thermal sensors using standard Fourier bases.

Chapter 4

Power Management Techniques

4.1 Introduction

One of the greatest challenges for today's cluster operators is the increasing energy cost as a fraction of the total cost of ownership. In fact, power and cooling costs have risen as much as 400 percent in the last decade [32]. Modern data center energy consumption results in millions of dollars in annual electricity costs in the U.S. alone. *Power capping*, in which the average or peak power of the cluster is constrained, has become a popular technique for ensuring energy budgets, planning cluster power delivery, and managing operational and cooling costs.

Wholesale energy markets introduce a new incentive for cluster power capping. Independent System Operators (ISOs) that coordinate power transmission have to match supply and demand in the grid. This challenge grows with the fluctuations in loads and sources as a larger portion of highly variable green energy sources are introduced into the grid [88]. As a result, many ISOs are looking into creating flexible reserves at the demand side. Large computing clusters are candidates for demand side regulation owing to the their load flexibility and power management features. ISOs offer credit to the demand side for regulating their power at several second intervals; thus, fine-grained modulation of cluster power has the potential to provide significant monetary savings.

Individual server power capping is an essential prerequisite to cluster-level power capping. Existing methods for server power capping include sleep modes, dynamic voltage-frequency settings (DVFS) [72, 110], low-power nap modes, and throttling by idle cycle insertion [35]. At a larger scale, it is possible to devise control techniques to coordinate multiple levels of capping in a data center [111], and to automatically adjust server caps based on utilization [4].

This chapter proposes a novel technique, *Pack & Cap*, for maximizing performance within dynamically set power caps for multi-threaded workloads. Pack & Cap builds on the observation that workloads on clusters are increasingly employing thread-level parallelism to capitalize on the hardware parallelism in multi-core processors. Parallel applications offer a new control knob for power capping, namely selecting the number of active threads. Pack & Cap leverages *thread packing*, where multiple threads of an application are packed onto a variable number of cores, as a low-cost proxy to mimic dynamic selection of active threads.

Pack & Cap brings several important innovations over the state-of-the-art. First, it is designed to meet instantaneous server power caps, while most prior techniques (such as throttling and DVFS) focus on maintaining an average power consumption value. Second, by controlling the number of active cores via thread packing in addition to DVFS, we are able to decrease the lower bound on achievable power caps and thus achieve more dynamic control flexibility. These two innovations enable a fine-grained power capping strategy and higher throughput, which make our technique attractive for use in dynamic power regulation. We propose a set of techniques for effective adaptive power capping. These techniques offer different tradeoffs in terms of effectiveness, hardware requirements, offline characterization efforts, and implementation costs. We conduct all experiments on a server with two quad-core processors, making our method attractive for deployment on real systems. We demonstrate that Pack & Cap meets the power caps with high accuracy, while minimizing the application runtime.

4.2 DVFS + Thread Packing

DVFS is a standard technique for dynamically enforcing power caps [111, 33]. A major contribution of our work lies in our use of a variable number of active cores as a control knob in conjunction with DVFS while running multi-threaded workloads. This additional control knob, which we call *thread packing*, allows us to (1) achieve more desirable power-performance tradeoffs compared to using DVFS alone, and (2) decrease the lower bound of dynamically achievable power caps, thus providing more flexibility for cluster-level capping strategies.

The inspiration behind thread packing comes from the power-performance tradeoffs observed while varying the number of threads in a parallel application, which we refer to by the term *thread reduction*. In our initial experiments, we investigate the effects of performing thread reduction, which we evaluate in terms of the achievable power levels with experiments on a multi-core server. Our server is equipped with two quad-core Intel Xeon E5520 processor chips (each core with 6 DVFS settings) and 12 GB of memory. In Xeon processors, idle cores automatically switch to low-power states through clock gating to save energy. We run each PARSEC benchmark [9] under all the DVFS settings with 1, 2, 4, and 8 threads. We then measure the power range, which is the difference be-

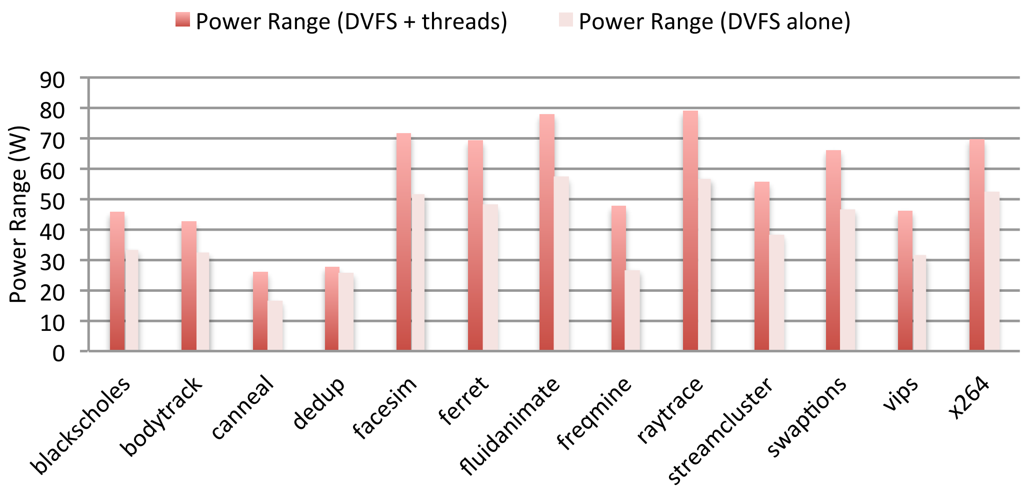

Figure 4.1: Dynamic power range achieved by DVFS and changing thread count compared to using DVFS alone. Experiments are conducted on a server with two quad-core Xeon processors.

tween the server's highest and lowest power level observed across all DVFS settings and thread counts. Without thread reduction, the lowest power level is achieved by setting the lowest DVFS setting. With thread reduction, however, lower power can be achieved by setting the lowest DVFS setting and reduction the number of threads. Idle cores entering low-power states yield significant power reductions. Thread reduction is useful when the lowest DVFS setting is insufficient for meeting a power cap. In our prior work, we showed that jointly controlling DVFS and the number of threads on a single-chip quad-core system increases the power range by 21% compared to using only DVFS on a single quad-core processor [18]. Further improvements in the power range are attainable in multi-processor systems, which are more typical configurations in today's clusters. The power range increases to 41% on average across all PARSEC benchmarks running on our dual-processor server. We display the power range comparison for each PARSEC workload for our dual-processor machine in Figure 4.1. In addition to the increased power range, integrating the use of the two knobs (e.g. DVFS, # threads) enables finer-grain capping, where power and performance can be tuned more precisely to meet desired constraints.

While thread reduction is an effective power management tool, it cannot be applied dynamically during execution without substantial modifications to the application code. Thread packing, on the other hand, is a practical alternative that is applied by modifying thread-core affinities in the operating system, thus eliminating the need for application-specific modification. Each multi-threaded workload is constrained to execute on a subset of available cores by setting thread-core affinities in the operating system (via the `sched_setaffinity` system call interface in Linux), while the remaining idle cores enter low-power states. Load balancing and scheduling of threads among the active cores on each processor are then performed transparently using the default operating system algorithms. To further motivate thread packing, we verify experimentally that each thread packing configuration has almost identical runtime and power characteristics to the thread reduction scenario with the matching number of active cores. We perform all experiments with the PARSEC benchmark suite [9] on the multi-chip 8-core server with static settings across the execution of each workload. We disable hyper-threading (SMT) in order to show the worst case performance penalty incurred by thread packing. We run each benchmark using the native input set at every DVFS settings with the following thread scenarios:

- **8 cores:** We measure power and runtime for 8 threads executing on 8 cores. This case serves as the baseline for both thread packing and thread reduction in the following three comparisons.
- **4 cores:** Power and runtime for executing 8 threads packed on 4 cores *against* executing 4 threads on 4 cores.
- **2 cores:** Power and runtime for executing 8 threads packed on 2 cores *against* executing 2 threads on 2 cores.
- **1 core:** Power and runtime for executing 8 threads packed on 1 core *against* executing 1 thread on 1 core.

We perform a comparison of runtime and average power between thread packing and thread reduction for the case of 1, 2, 4 and 8 active cores for all of the PARSEC benchmarks. For each comparison, the number of active cores is the same for both thread packing and thread reduction (e.g., 8 threads packed onto 2 cores compared to 2 threads running on 2 cores). Packing and reduction perform identically in the 8-core case. For the 4-core case, thread packing increases runtime by 3.6%, but decreases average power consumption by 0.1%. For the 2-core case, thread packing decreases runtime by 4.5% and decreases average power by 0.9%, and for the 1-core case, packing increases runtime by 0.7% and decreases the average power by 1.5%. These results show that when fully utilized, the number of active cores is the primary determiner of power and runtime for a multi-threaded application. While the runtime and power values are comparable, thread packing permits for dynamic adjustment during workload execution without workload specific modifications. In addition, the allowable number of active cores in thread packing is not constrictive. Many of the PARSEC benchmarks can only be launched with thread counts that are a power of two. Thus, thread packing is a more practical and flexible solution to dynamic power capping.

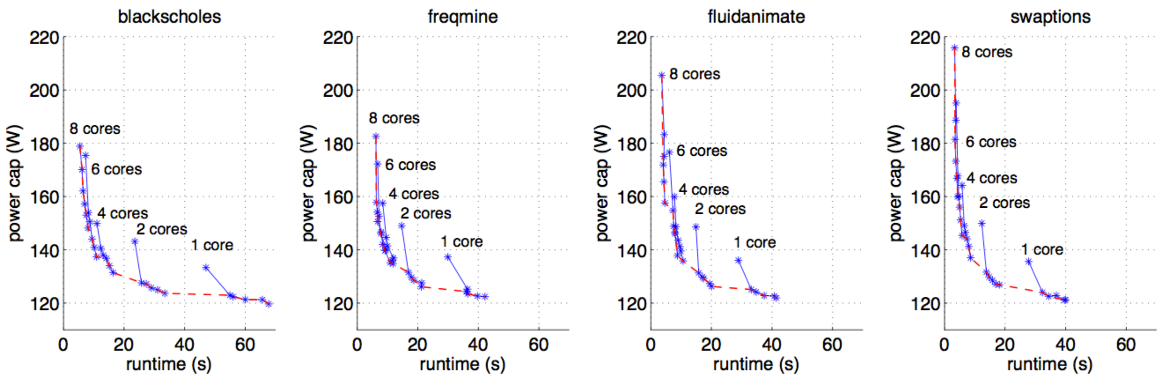

Figure 4.2: Plots show the impact of DVFS and thread packing settings on runtime and power consumption for dual quad-core processor machine. Red line gives Pareto frontier of optimal settings at various power caps. Each blue line gives the power and runtime results when we change the DVFS under a fixed number of cores. We include only the cases of 8, 6, 4, 2, and 1 cores. We leave out the 3, 5, and 7 core cases for the sake of clarity.

When minimizing runtime in the absence of power caps, the optimal operating point is trivially the one with the maximum number of active cores and maximum DVFS setting. In the presence of power caps, however, the optimal setting that produces the best performance within a fixed power cap varies depending on workload and environmental conditions. Figure 4.2 provides the peak power and runtime at all possible settings for the first 100 billion retired μops of four PARSEC benchmarks on our dual-processor server. To simplify the task of finding for the optimal setting, we mark the *power-runtime Pareto frontier* with a red line in Figure 4.2 given power and runtime measurements for each setting. For each point along the frontier, there is no alternative point that achieves lower peak power *and* shorter runtime. Any point that is not on the frontier cannot be optimal because there exists a setting that on the Pareto frontier that produces both lower runtime and lower peak power. Thus, the points on the frontier *dominate* the non-frontier points, and the point along the frontier with the least runtime within the power cap is optimal. We observe that for all frontiers, within the subset of points for which the power cap is met, the optimal point that minimizes runtime always maximizes the number of active cores. Thus, any control policy should select the highest number of cores for which the power cap can be met first, and then select the highest DVFS setting within the power cap.

It is worth noting that our experiments with thread packing SPEC CPU2006 [106] workloads show that this observation applies for single-threaded applications as well. The combined throughput of a set of single-threaded applications scales well with the number of active cores, which is intuitive, as single-threaded applications are not limited by the synchronization overheads encountered in multi-threaded applications. We have also characterized the behavior of the PARSEC workloads with hyper-threading enabled and seen that our observations about the Pareto frontier still hold. That is, the optimal operating point will always maximize the number of active physical cores within the power constraint.

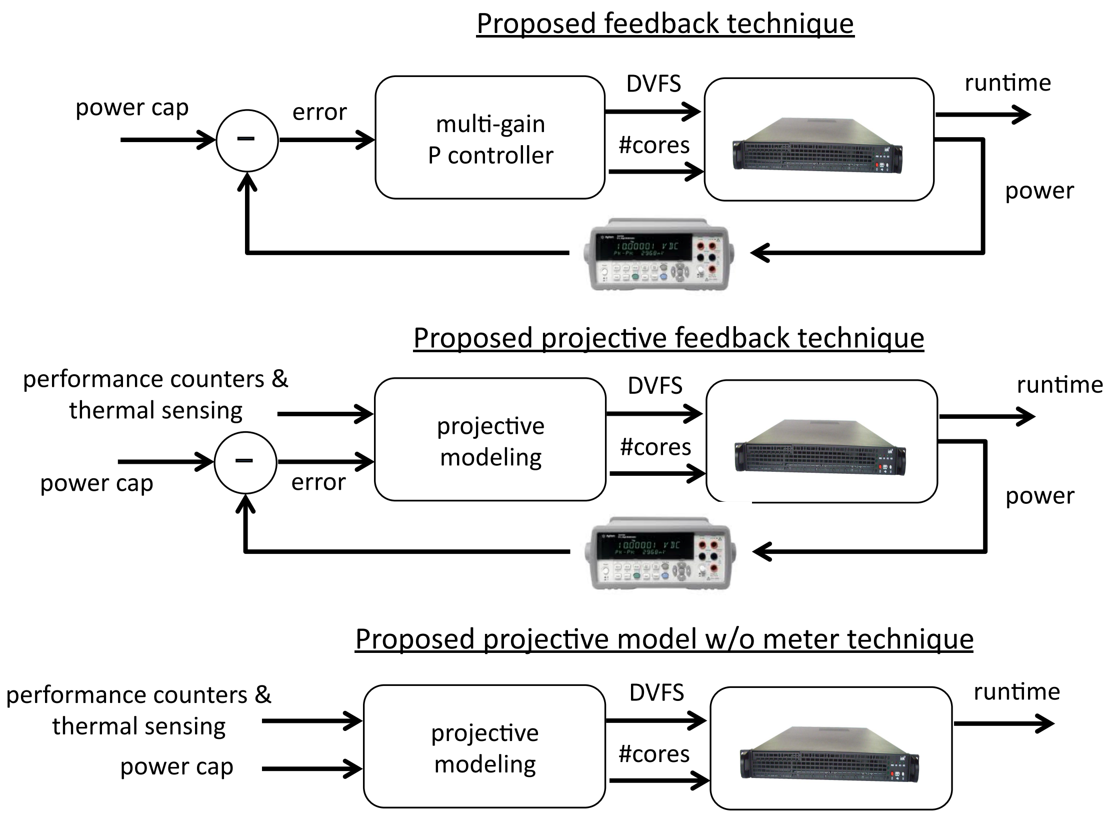

Figure 4.3: Proposed power capping methods.

4.3 Pack & Cap Methodology

In this work, we propose *three techniques* for minimizing application runtime within a power cap. These techniques, illustrated in Figure 4.3, offer different tradeoffs in terms of effectiveness, hardware requirements, and implementation costs. We propose a *feedback technique* with a multi-gain controller that utilizes runtime power measurements without requiring any measurements of server state (e.g., performance counters, temperatures, or utilization). We then propose a *projective feedback technique* that improves capping accuracy and performance by leveraging measurements of the server's state and extensive offline characterization. The third technique, *projective modeling w/o meter*, is similar to the second technique, but it does not use a power meter. Projective modeling avoids the additional equipment cost of power telemetry at the expense of power capping accuracy.

1. Proposed Feedback Technique. Feedback techniques assume that a server has access to power measurements through either an integrated or external power meter. Based on the measured *power slack*, i.e., the difference between the required power cap and the actual power consumption, the feedback controller chooses appropriate settings to reduce the slack. Classical feedback-based capping methods solely use DVFS for feedback control, where the DVFS setting is adjusted using a P or PI controller based on the power slack [72, 110]. In addition to DVFS, our technique adjusts the number of active cores (i.e., thread packing configuration). The main challenges are (1) choosing between the two control knobs to meet the power cap, and (2) tuning the controller to achieve the desired balance between response time and meeting the cap.

To choose among the two settings, we propose a *heuristic* that is based on the Pareto frontier characterization results of Section 4.2, in which we observe that within the subset of points that adhere to the power cap, the optimal point along the Pareto frontier maximizes the number of active cores. It is therefore best to avoid decreasing the number of cores in order to reduce negative power slack, and to increase the number of cores whenever possible to eliminate positive slack. Our heuristic makes use of experimentally calculated proportional power gains in order to estimate the effect of DVFS and the number of active cores on power. If a positive power slack is observed, the controller increases the number of active cores to the maximum number estimated to be within the power cap. The remaining slack is then reduced by increasing the DVFS setting in the same manner. If a negative power slack is observed, then the controller selects the highest DVFS setting estimated within the power cap. If the minimum DVFS setting is estimated to be above the power cap, however, then the number of active cores is decreased according to its proportional gain to be within the cap. In this way, the controller prioritizes thread packing (by increasing number of cores) over DVFS when there is a positive power slack, and prioritizes DVFS over packing when there is a negative power slack. In our implementation, we measure the power slack and apply feedback at each activation (e.g., every 1 second) of the controller.

During offline characterization of power and runtime, we observe that DVFS proportional gain depends on the number of active cores, and similarly, the thread packing proportional gain depends on the DVFS setting. Making use of this observation, we propose the use of a *multi-gain controller* in which the proportional gains for both control knobs are selected according to the current setting. We use a set of offline characterization data gathered across multiple workloads to calculate the average change in power per unit change in each control knob (e.g., DVFS setting or #cores), while the other control knob (e.g., #cores or DVFS) is held constant. A different gain is calculated for each value of the static control knob. Thus, we calculate a thread packing proportional gain for each DVFS setting, and a DVFS proportional gain for each thread packing configuration.

2. Proposed Projective Feedback Technique. The feedback technique is able to successfully maintain a power cap. However, it does not always yield the minimum application runtime within the given cap as it uses a heuristic to explore the Pareto Frontier. In addition, while the multi-gain controller in the feedback technique differentiates gains based on the current DVFS and number of active cores, it does not explicitly model different workload characteristics. We improve the feedback controller by incorporating a workload-sensitive *projective modeling* technique that estimates the projected power of all possible DVFS and thread packing settings on the power slack as a function of *state measurements* (i.e., performance counter and core temperature measurements), and then selects the setting projected to be within the power cap that has the least runtime. This modeling-based approach for adaptive power capping is divided into an offline and online phase. The offline phase is computationally demanding and it is only performed once for a particular server configuration. The results of the offline phase are stored in look-up tables that are accessed by the online phase. The online phase is computationally lightweight and can perform adaptive power capping on any server that has the same hardware configuration as the training server.

In the offline phase, we use an extensive set of data collected for multi-threaded parallel workloads (e.g., PARSEC benchmark suite) to train separate power estimation models for each DVFS and packing setting. Individualizing the models in this way emphasizes the salient characteristics at each control setting. The training data consists of per-core temperature measurements, system power measurements, and performance counter measurements, which include the number of μops retired, floating point operations, load locks, resource stalls, branch prediction misses, L2 misses, and L3 misses (to capture main memory activity). It is worth noting that because these metrics are gathered per-core and not per-workload, any model that takes them as input is *globally* defined and will not change depending on the workload. Our observations about performance counters indicate that much of the variation in power and delay among workloads can be attributed to memory-boundedness (i.e., the ratio of memory access to instructions executed) as noted in prior work [51]. For more memory-bounded applications, the sensitivity of power and runtime to DVFS and thread packing is much lower, as the workload incurs fixed latencies while stalling for cache misses and memory accesses. For less memory-bound applications the change in both power and runtime is much higher. For a frequency/thread packing setting (f, t), a power estimate, $\hat{P}_{f,t}[k]$, at time instant k is given by

$$\hat{P}_{f,t}[k] = \mathbf{c}_{f,t} \cdot \mathbf{x}_{f,t}[k], \tag{4.1}$$

where $\mathbf{x}_{f,t}[k]$ denotes the input vector of state measurements, $\mathbf{c}_{f,t}$ is the vector containing model coefficients, and the operator $\cdot$ denotes the dot product operation. Note that the input vector $\mathbf{x}$ includes a constant term in addition to the state measurements (16 terms total). Such regression models have been successful in the past [67].

For learning the model coefficients, we use *robust regression* to reduce the impact of spurious outlier measurements (e.g., from operating system calls) on the learned models. Robust regression seeks to minimize a weighted total square error, i.e., $\sum_k w_k e_k^2$, where $e_k = \hat{P}_{f,t}[k] - P_{f,t}[k]$, in which $P_{f,t}[k]$ is the true power consumption [2]. To discard outliers, the weights, w_k, should increase as the error residual $|e_k|$ decreases in value. A popular weighting function we use is the bi-square function. In this function, $w_k = 0$ when $|e_k| > r$, and $w_k = (1 - (e_k/r)^2)^2$ when $|e_k| <= r$, where r is a constant that determines the extent of outlier rejection.

Because the input metrics in $\mathbf{x}$ are themselves dependent on the control setting (f, t), it is necessary to multiply the inputs by the *mapping ratios*. These ratios *project or map* the measurements of the performance counters at the current setting (i, j) to any other candidate setting (f, t); i.e., $\mathbf{x}_{i,j} \xrightarrow{m} \mathbf{x}_{f,t}$. Ratios are learned from the offline characterization data by computing the expected measurement values of performance counters at every setting combination. If $E[\mathbf{x}_{i,j}]$ denotes a vector of expected input values at setting (i, j), then the multiply ratios for mapping the input state from setting (i, j) to (f, t) is $E[\mathbf{x}_{f,t}]/E[\mathbf{x}_{i,j}]$ (element-wise division). These mapping ratios together with the model coefficients and error standard deviations for the model learned at each setting are stored in lookup tables for use at runtime.

At runtime, our online power capping system logs performance counter and temperature data periodically, and identifies the optimal operating settings. If measured power at time k and current setting (i, j) is denoted by $P_{i,j}[k]$, then the power slack is denoted with $\delta[k] = P_{cap}[k] - P_{i,j}[k]$. Given the current input measurement vector $\mathbf{x}_{i,j}[k]$, the capping policy first identifies a set S of $\{$DVFS, #cores$\}$ settings such that the projected power consumption from any setting $(f, t) \in S$ is the within the power cap. That is,

$$S = \{(f,t) | \text{ where } \hat{P}_{f,t}(\mathbf{x}_{i,j}[k] \xrightarrow{m} \mathbf{x}_{f,t}[k+1]) \leq \hat{P}_{i,j}[k] + \delta[k]\}, \tag{4.2}$$

where $\hat{P}_{i,j}[k]$ is the current power estimate. Note that $\hat{P}_{i,j}[k] + \delta[k]$ is equal to $\hat{P}_{i,j}[k] + P_{cap}[k] - P_{i,j}[k] = P_{cap}[k] + (\hat{P}_{i,j}[k] - P_{i,j}[k])$. The term $(\hat{P}_{i,j}[k] - P_{i,j}[k])$ measures the projective model error and converges to 0 when the modeled power matches the measured power. In this way, power measurement feedback is used to increase robustness against constant offset modeling errors. Constant deviations between the model estimates and the measured power can arise when the test system differs from the offline characterization system (e.g., due to process variations, slightly different hardware configuration, ambient temperature variation, etc.). In Equation (4.2), the state measurements $\mathbf{x}_{i,j}[k]$ at the current setting (i,j) are first mapped into each potential new setting (f,t), and then the power consumption of each mapped measurement is estimated using the power regression model $\hat{P}_{f,t}(\cdot)$. To pick the optimal setting, $(\hat{f},\hat{t}) \in S$, the controller first filters S to retain the settings that use the largest number of cores $\hat{t}$ and then chooses the highest DVFS setting $\hat{f}$ within the filtered set. This selection process follows the Pareto frontier observations for minimizing runtime as discussed in Section 4.2. The projected optimal setting is then applied to the server as shown in Figure 4.3. The overall runtime complexity of online power capping with this approach is linear with the number of control settings, which promises to scale well to larger numbers of settings in many-core architectures.

3. Proposed Projective Model Without Meter Technique. In this technique, we assume that the server does not have the ability to measure power consumption. This situation can arise with old servers or more economical servers that do not include power sensing equipment. For such cases, we only use the model estimate $\hat{P}_{f,t}(\cdot)$ from the *projective feedback* technique to guide setting selection. That is,

$$S = \{(f,t)| \text{ where } \hat{P}_{f,t}(\mathbf{x}_{i,j}[k] \xrightarrow{m} \mathbf{x}_{f,t}[k+1]) \leq P_{cap}[k]\}, \tag{4.3}$$

and the optimal setting is selected from S as discussed earlier. In this case, power measurements are only used for offline learning but are not incorporated into runtime control. The main drawback of this approach is that constant offset modeling errors translate directly to power cap violations. However, our experimental results demonstrate that this method can achieve relatively good results despite the lack of hardware power telemetry. In addition, if power cap violations are normally distributed with no bias for positive or negative power slack, then the technique can be successfully leveraged for capping average power.

4.4 Experimental Results

We evaluate the Pack & Cap methodology in terms of the accuracy in adhering to dynamic power caps and application performance. For quantifying the accuracy, we measure the *power cap error*, which is the average negative power slack magnitude normalized by the power cap value. The performance is simply measured using workload runtimes. All of the experiments are performed on dual Intel quad-core Xeon E5520 system. Each processor has 6 DVFS settings ranging from 1.60 GHz to 2.27 GHz in 0.13 GHz increments. We disable the hyper-threading feature. The server runs a Linux kernel 2.6.10.8 OS. We use `pfmon` to collect performance counter data and `lm-sensors` to poll the on-chip thermal sensors. We measure the server's total power consumption using an Agilent 34410A digital multimeter.

For comparison, we implement a *baseline feedback technique* that incrementally adjusts both the DVFS setting and the number of active cores (i.e., reduces setting by one increment for negative power slack, increases by one increment for positive slack). This technique engages DVFS first in order to meet the power cap and only adjusts the number of active cores when limited by the maximum or minimum frequency. This policy is a natural extension of the DVFS feedback techniques used in previous works for meeting power caps [72]. By comparing to the baseline technique, we quantify the benefits of incorporating our Pareto frontier observations into our proposed control techniques.

For all experiments, control is activated once per second. The time overhead for calculating control decisions for all Pack & Cap techniques is on the order of 10 ms, which is less than 1% of the control activation period. The time overhead for performing DVFS control is on the order of microseconds. While the overhead for shifting threads among the cores in thread packing is potentially higher, all performance overheads are automatically accounted for in our workload runtime results. For all feedback techniques, we avoid any actuation if the power consumption is within 2 W below the power cap because our offline characterization shows that setting changes trigger power changes larger than 2 W. We apply the same rule for the projective modeling technique without meter, except that the power consumption is determined through modeling rather than measurement.

To evaluate the effectiveness of the proposed techniques in a realistic situation in which data center nodes must meet dynamic power caps requested by ISOs, we set an aggressive dynamic cap that changes every 10 seconds to a random value in the range 120 W - 170 W for each PARSEC workload running with 8 threads. Figure 4.4.a and Figure 4.4.b compare the runtime and capping accuracy of each workload. The runtime values are normalized to those observed for the baseline feedback technique. The three proposed Pack & Cap techniques outperform the baseline technique significantly in both runtime and power cap accuracy. The baseline feedback controller delivers larger power cap errors (average 7.5%) with larger runtimes, as it does not utilize power slack magnitude or knowledge of the Pareto Frontier. Our feedback technique and projective feedback technique deliver the best results in terms of accuracy, with average cap errors of 3.5% and 3.6% respectively. The

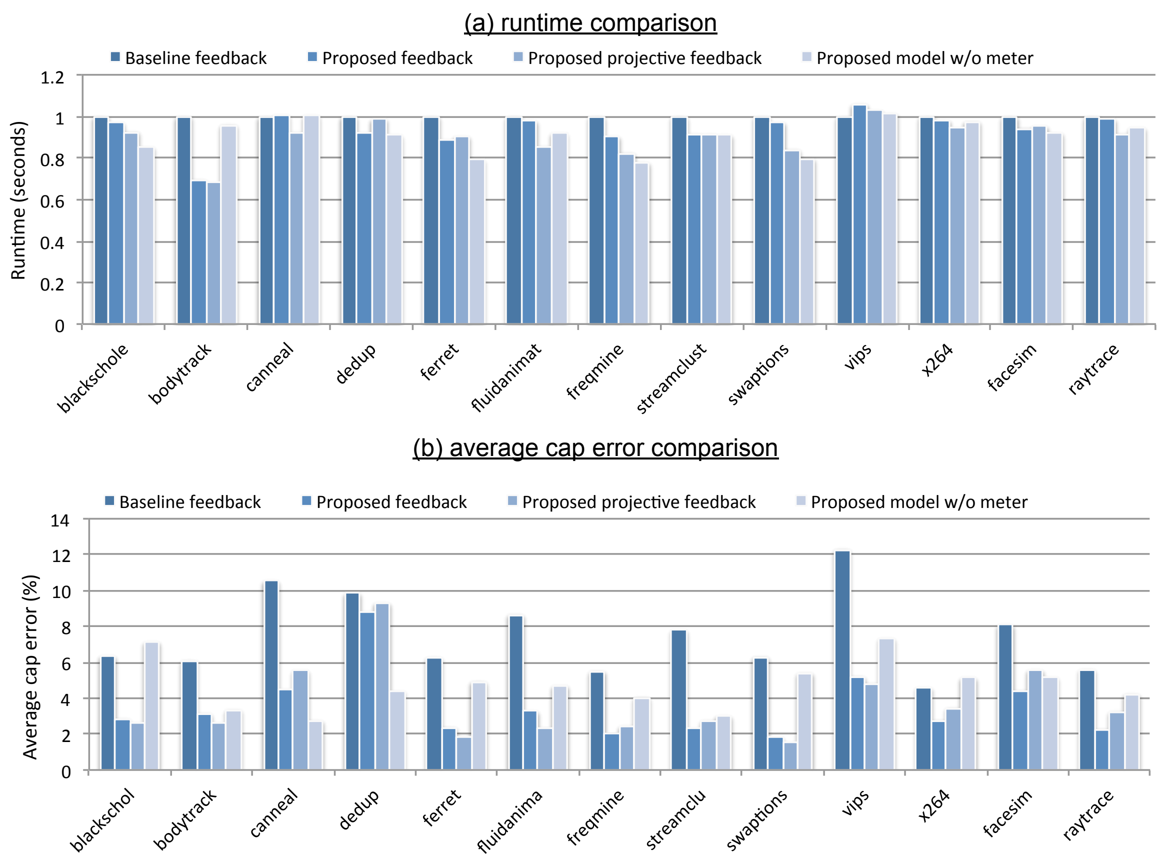

Figure 4.4: Runtime and average power cap tracking accuracy of proposed techniques.

projective feedback technique outperforms the feedback technique in runtime with 0.90× normalized runtime compared to 0.94×. This improvement is a result of the projective feedback technique's superior Pareto Frontier tracking ability.

The results in Figure 4.4.b also show that, as expected, the projective model w/o meter technique delivers worse power capping accuracy, with an average error of 4.73%. The runtime of this approach is comparable to the projective feedback technique with 0.91× reduction over the baseline. Without using measurement feedback, the projective model w/o meter technique has no way to correct for modeling errors. These errors can become significant if the system differs between online testing and offline characterization, or if the model poorly captures a particular workload. For instance, `bodytrack` shows larger runtime and worse power capping accuracy compared to the other proposed techniques,

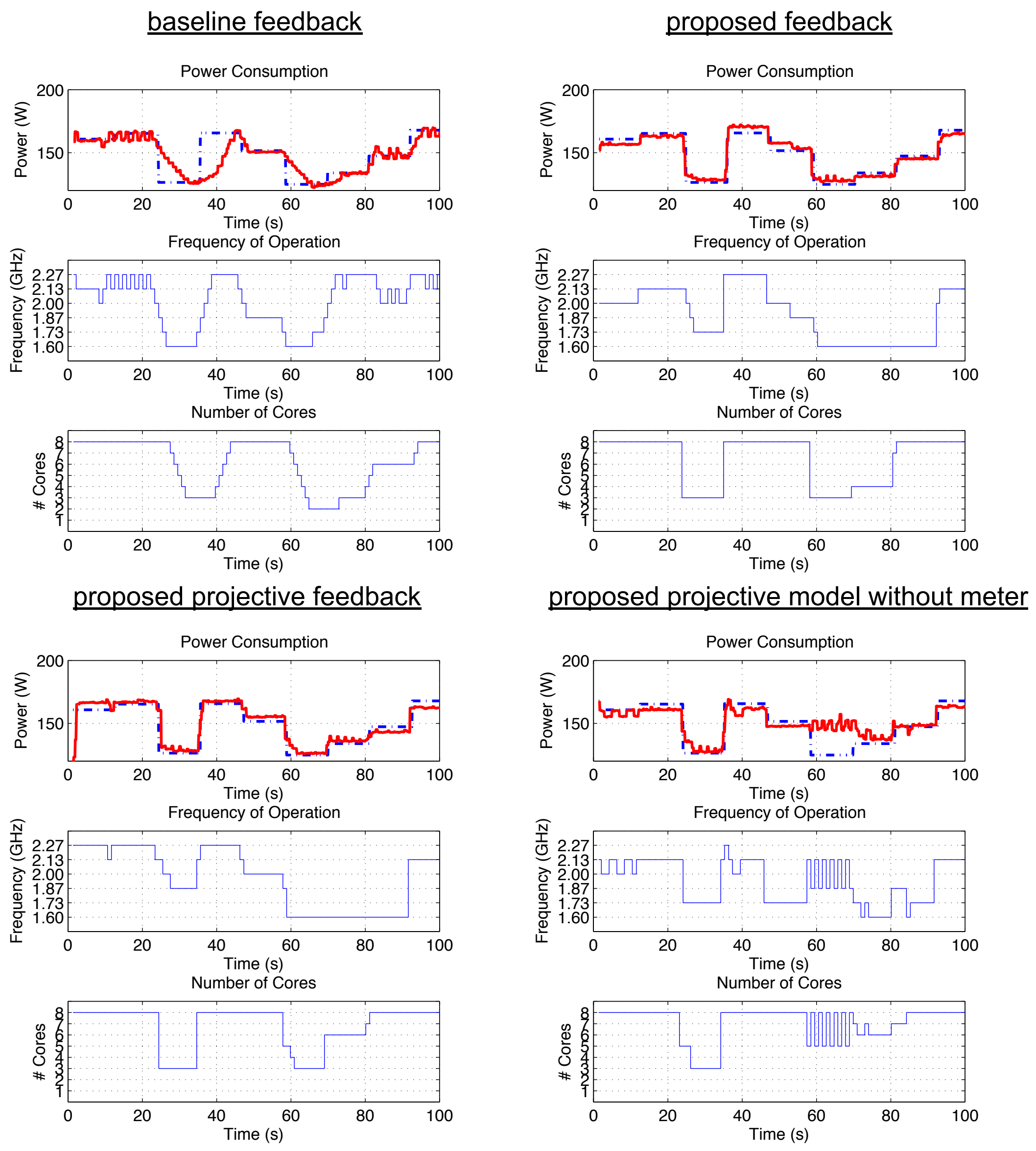

Figure 4.5: A detailed exploration into the first 100 seconds of the `blackscholes` application, demonstrating the selected DVFS and thread packing settings together with system power consumption.

indicating power over-estimation and under-estimation. Nevertheless, the reasonable accuracy result indicates that this technique is a viable option for power capping without power measurement equipment. Feedback-based techniques eliminate such deviations at the expense of incorporating hardware power meters.

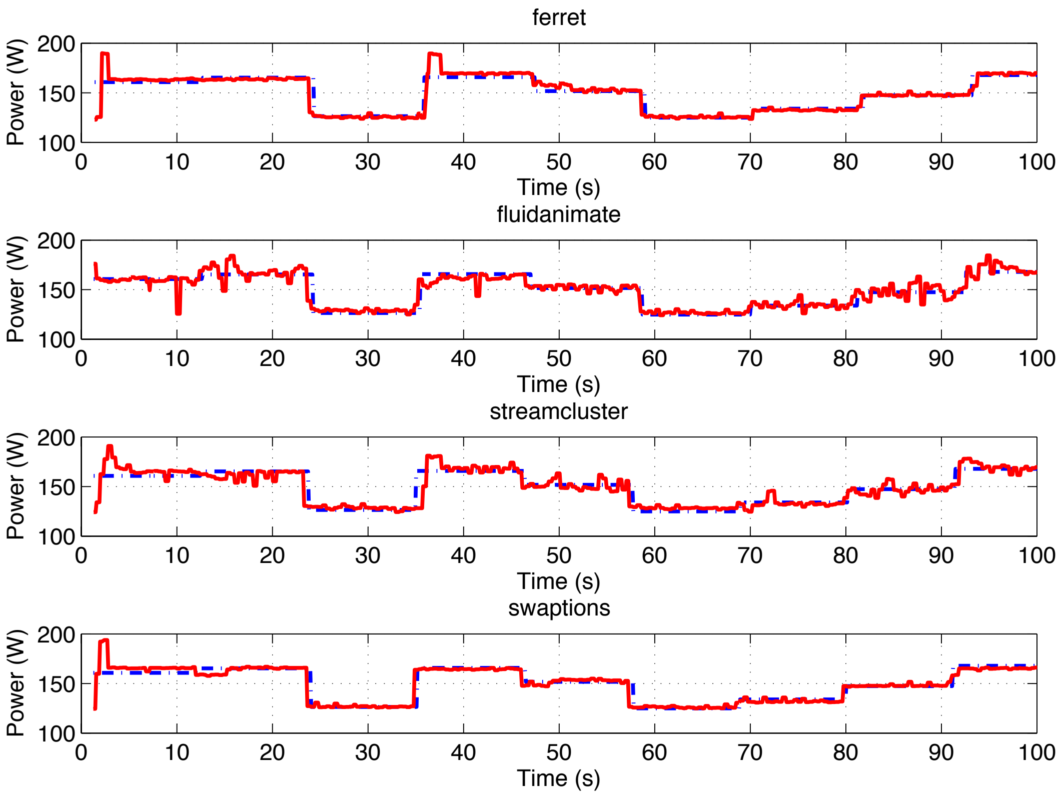

Figure 4.6: Measured power traces for `ferret`, `fluidanimate`, `streamcluster`, `swaptions)` benchmarks as Pack & Cap adapts performance to constrain the power consumption with the power cap (blue dotted line). The power cap is randomly modulated every 10 seconds.

Figure 4.5 illustrates the measured power consumption and the control decisions (DVFS, #cores) for the `blackscholes` benchmark over time. The four methods in order of presentation in Figure 4.5 have an average setting of (1.85 GHz, 6.0), (1.93 GHz, 6.3), (1.92 GHz, 6.6), and (1.92 GHz, 7.1) respectively. Note that the projective feedback technique meets the power cap using a higher number of cores and lower average DVFS setting than the feedback technique, which clearly indicates superior Pareto Frontier tracking. All of the proposed techniques track the power cap tightly, eliminating power slack and reducing runtime relative to the baseline feedback technique. The disadvantage of the projective model w/o meter technique can be clearly seen in 60 - 70 second interval. The power

	baseline feedback	proposed feedback	proposed projective feedback	proposed projective model without meter
Hardware requirements	power meter	power meter	power meter performance counters thermal sensors	performance counters thermal sensors
Offline characterization	none	simple	intensive	intensive
Power cap error	7.52%	3.51%	3.69%	4.73%
Application runtime	1.00×	0.94×	0.90×	0.91×

Table 4.1: Summary comparison among the proposed Pack & Cap methods and baseline method.

projected by the model deviates from the measured power, leading to significant power capping error. However, the measurement feedback in the projective feedback technique successfully eliminates this modeling error.

To illustrate power track further, we also plot the power consumption of `ferret`, `fluidanimate`,`streamcluster`, and `swaptions)` using the same sequence of dynamic power caps as before in Figure 4.6. We use the projective feedback method as the capping policy. Despite major differences among application characteristics, our technique is able to track the dynamic power caps with high accuracy where the power is within an average value of 4.7% of the cap.

We summarize the tradeoffs between our proposed techniques in Table 4.1. We have also implemented our modeling without meter approach on a single-processor server and compared it to our earlier capping work, which uses multinomial logistic regression (MLR) for classification and does not use power meters [18]. Projective modeling without meter shows a consistent improvement of 20% to 30% in runtime while still meeting the power caps with the same accuracy as the MLR classifier.

4.5 Summary

Power capping is gaining importance as a method to manage energy costs and plan power delivery in large computing clusters. In this chapter, we introduce thread reduction for multithreaded workloads as a means of meeting low power budgets on a single server node, thus increasing power allocation flexibility for datacenters composed of many server nodes. Because thread reduction is not easily performed dynamically during runtime, we propose thread packing, in which multi-threaded workloads are packed onto a variable number of active cores, as a proxy for thread reduction. We show that thread reduction and thread packing have nearly identical power and performance characteristics. We then introduce Pack & Cap, which makes optimal DVFS and thread packing control decisions such that performance is maximized within a power budget. We illuminate power and performance trade-offs between thread packing and DVFS and develop heuristics for navigating Pareto efficient control settings. We implement Pack & Cap with several candidate power models, including a multi-gain feedback controller as well as open-loop and closed-loop linear regression models. We compare these techniques to a baseline feedback technique and demonstrate up to 10% reduction in average runtime and up to 53% improvement in average power cap accuracy. We also compare to our previous work in [18], which uses a multinomial logistic regression (MLR) classifier, and show up to 30% runtime improvement with similar power cap accuracy. Our next steps include extending the Pack & Cap methodology to heterogeneous architectures, and integrating it within group power capping schemes in which sets of nodes are managed together under a common power constraint. We will also expand our techniques to differentiate between critical and non-critical threads when performing thread packing (e.g., [8]).

Chapter 5

Thermal Management Techniques

5.1 Introduction

Temperature has become a true limiter to the performance and reliability of computing systems. The recent emergence of temperature as a fundamental bottleneck is a consequence of continued ideal geometric scaling and suboptimal electric scaling. Less than ideal scaling of supply voltages and threshold voltages have created a situation in which leakage and dynamic power are not keeping pace with geometric scaling. Many-core architectures lead to localized temperature hot spots that severely limit overall system performance, as the speed of transistors and interconnects are negatively affected by temperature [11]. In addition, elevated temperatures cause circuits to deteriorate structurally. All circuit breakdown phenomenon (e.g., electromigration, time dependent dielectric breakdown, and negative bias temperature instability) are highly temperature dependent [89], and thermal cycles create mechanical stresses due to expansions and contractions [11].

Every processor has a temperature limit above which the risk of physical damage increases significantly. In response to thermal emergencies in which the processor die temperature is in danger of exceeding this level, most processors will automatically shut down for protection. In high performance processors, this temperature limit is the primary constraint on performance, and the performance potential of such processors increases with the system cooling capacity [38]. In the past, cooling systems were designed such that a processor could never reach the upper limit under normal operating conditions. As thermal safety was effectively guaranteed, processors did not need to protect themselves. For modern processors, traditional air-cooling systems are unable to handle the worst-case power dissipation. With higher power densities and increasingly complex designs, it is no longer cost effective to design advanced cooling systems to handle the worst-case thermal emergency. Instead, modern cooling systems are designed to handle typical behavior, and processors must transparently protect themselves against the rarely occurring emergency by assessing the thermal risk throttling the performance accordingly.

Dynamic thermal management (DTM) techniques allow processors to assess thermal risk and control temperature. The most well-known DTM control "knobs" include dynamic voltage frequency scaling (DVFS), clock gating, and thread migration/scheduling [115, 44, 38, 25], which allow the processor to throttle performance levels and optimally distribute workloads. In addition, DTM techniques include control algorithms that use knowledge about the system state in order to manage the tradeoff between performance and thermal risk. A typical control scheme has a series of thermal thresholds associated with progressively harsher performance penalties approaching the thermal emergency level. The lowest threshold corresponds to a temperature set-point, or *soft threshold*, under which the processor will try to maintain its temperature using fine-grained control changes, such as selecting the optimal DVFS setting. Violations of the set-point incur only incremental performance penalties. Each threshold above the set-point, or *hard thresh-*

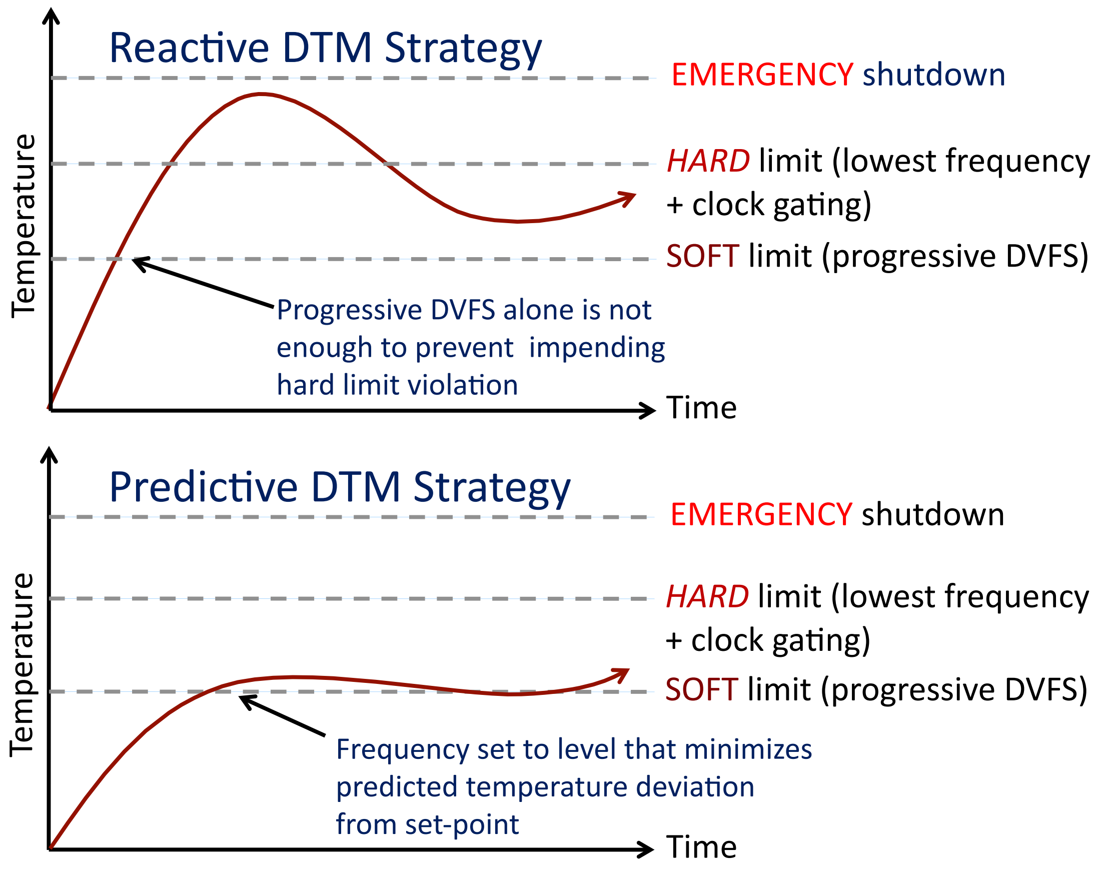
Reactive DTM Strategy
EMERGENCY shutdown
HARD limit (lowest frequency + clock gating)
SOFT limit (progressive DVFS)
Progressive DVFS alone is not enough to prevent impending hard limit violation
Temperature
Time
Predictive DTM Strategy
EMERGENCY shutdown
HARD limit (lowest frequency + clock gating)
SOFT limit (progressive DVFS)
Frequency set to level that minimizes predicted temperature deviation from set-point
Temperature
Time

Figure 5.1: Illustrations of the DTM control strategy employed by the Corei7 processor compared to a predictive approach.

olds, is associated with progressively harsher performance penalties as the thermal risk grows. For instance, many commercial processors use clock gating in addition to the lowest DVFS setting when the temperature exceeds a hard limit, which translates into a very harsh performance penalty. If the penalties associated with the hard temperature limits fail to constrain the temperature, it eventually reaches the emergency threshold and the system shuts down in response. This distance between the set-point (soft limit) and the emergency level constitutes the *thermal guard band*. The upper portion of Figure 5.1 illustrates a sample scheme employed by many popular commercial processors [7], where the processor incurs harsh clock-gating penalties when the temperature exceeds the hard limit.

The thermal response to control decisions, or *control response*, is highly workload dependent. Workload characteristics are complex and impossible to completely anticipate, and yet processors must guarantee thermal safety for the entire range behaviors. Commercial processors typically manage this complexity with reactive feedback techniques. If the temperature exceeds thermal thresholds, the system detects it with thermal sensors and scales back chip power consumption using the available control knobs. In recent academic works, it is commonplace to augment thermal sensor information with a thermal model that anticipates the thermal control response [11, 112, 102, 120, 65, 109, 25, 23, 63, 69, 117]. By increasing the certainty about the temperature behavior in response to control decisions, thermal modeling increases the performance potential of a device in two ways. First, for a fixed hard-limit violation rate, the more accurate thermal prediction strategy can have a smaller thermal guard band, thus permitting higher performance levels. Second, when holding the thermal guard band constant, a predictive strategy incurs fewer hard threshold violations and associated performance penalties.

In order to realize this performance potential however, these thermal models must be sensitive to differences in workload behavior; The thermal response to control decisions changes significantly as a function of the executing workload(s). The discrete time-invariant state-space model in Equation 2.1 is a standard model used for a variety of runtime control objectives in the literature [11, 112, 102, 120, 65, 109]. The workload thermal contribution is a function of the system control setting s as well as a vector of workload metrics $\mathbf{x_m}$ for core m. Capturing changes in heat source magnitude and location as a function of workload characteristics is a difficult task. A detailed model requires knowledge of the processor floorplan and numerous assumptions about the relationship between workload behavior and functional unit power consumption, all of which must be manually validated. In addition, it must handle the temperature dependence of leakage power consumption. Any manual validation performed for one chip design does not extend to future designs in which the physical layout and functional unit power characteristics change significantly. This work makes the following contributions in solving this problem:

1. We introduce a self-contained, data-driven, and fully automated method for capturing the workload dependence of $\widehat{g}(\cdot,\cdot)$ in Equation 2.1 with appropriate granularity using workload phase detection. We define workload phases as a function of processor *performance counters*, which measure architectural events (instructions executed, cace-misses, branch mis-predictions, etc.). These workload phases are *globally defined* in that they are not associated with any particular workload or mix of workloads. Using thermal sensor and performance counter measurements across a range of control settings and workload behaviors, we use classification techniques from the field of machine learning to automatically discriminate different control responses and associate them with workload behaviors. By learning the thermal response for each workload phase offline, we reduce the runtime overhead to a handful of linear combination calculations.

2. We validate our technique entirely on a real quad-core Intel Core i7 940 based workstation running heterogenous workload mixes selected from the SPEC CPU2006 benchmark suite. Our implementation uses real temperature values measured by per-core embedded sensors, and manages the workstation thermal behavior through DVFS control.

3. While our model is viable for many of the control objectives in previous works, we demonstrate the utility of our thermal prediction methodology in a proactive DVFS control scheme that is capable of maximizing performance within a temperature constraint. It is worth noting that we do not consider the thread packing control knob introduced in Chapter 4 for DTM because the temperature constraint does not changed as a part of a larger control objective, as is the case with server

power capping. A higher temperature threshold is associated with a higher power consumption, and we show in our Pareto frontier characterization in Section 4.2 that thread packing is only advantageous for low power caps. As a result, it is unlikely that a thermal threshold would ever be low enough to necessitate thread packing. In comparison to our previous work on workload phases [22], we show a 0.99% increase in instruction throughput and a 60% reduction in thermal violations. When we compare our thermal phase classification techniques to the sequential-probability-ratio-tests (SPRT) used for switching thermal models in previous works [23], we demonstrate a 2.9% improvement in instruction throughput and a 97% reduction in thermal violations. In comparison to state-of-the-art model predictive control (MPC) techniques [5, 6], we demonstrate a 5.8% improvement in instruction throughput with the same number of thermal violations. Finally, we compare to a simple proportional-integral (PI) technique and demonstrate a 3.9% throughput improvement with a 94% reduction in thermal violations.

The rest of this chapter is organized as follows. We provide an overview of our methodology in Section 5.2. We go on to describe our phase classification approaches in Section 5.3, and in Section 5.4 we describe our predictive DVFS control strategy. We provide a comprehensive set of experimental results on a real quad-core system in Section 5.5. Finally, Section 5.6 summarizes the main conclusions drawn from this work.

5.2 Methodology

At the highest level our thermal prediction and control strategy divides between offline analysis and runtime control. We avoid the computational cost of online learning in previous works [23, 118] by learning our models offline with training data that spans a large range of workload behaviors. We then validate our models at runtime using workload mixtures unseen in the training data. During offline analysis, the complex relationships between die temperatures, workload behaviors, and processor frequency are learned using an extensive set of training samples. The training data is comprised of per-core temperature and performance counter measurements taken across a range of workloads, processor utilization levels, and DVFS settings. Taken as a whole, the performance counters expose the architectural behaviors associated with workload phases, which we define in this context as periods of workload execution that demonstrate a consistent thermal control response. In order to define phases, we develop a phase classifier that estimates the likeliest phase as a function of performance counter inputs. Each phase is then associated with a thermal control model that predicts each core's thermal response to DVFS control. During runtime, the phase classifier identifies the current workload phase for each core, and the corresponding thermal model is used by the DVFS controller. By having advanced knowledge of the thermal trajectory, the controller is better able to maximize performance and minimize thermal risk. We explore several methods for phase classification, which we detail in Section 5.3. While the offline analysis is computationally intensive and requires an extensive set of training samples, runtime application is a simple model query and incurs minimal performance overhead.

The overall goal is to develop a modeling technique that captures the complex relationship between chip temperature and the processor frequency as a function of the workload behavior. This workload dependence is nontrivial, as different workload behaviors produce different spatial distributions of power at the microarchitectural level, even if the total power dissipation is the same. For instance, a workload that exercises the floating point unit or branch prediction unit intensely will register higher temperatures than a workload that exercises the caches, as the power dissipation is concentrated in a much smaller area. In addition, the relationship between the maximum temperature and architectural behavior can change depending on the location of the maximum temperature hot spot. If the largest hot spot is located on the floating point unit, then the maximum die temperature will be a strong function of the number of floating point operations. However, if the hot spot is elsewhere on the chip, then the maximum temperature could be entirely independent of floating point activity. The situation is further complicated by the fact that dynamic power dissipation is somewhere between being quadratically and cubically dependent on the processor frequency, depending on how the voltage scales, and by the leakage power dependence on temperature.

There are two possible approaches for modeling this complexity: develop a single complex model that is workload dependent and encapsulates nonlinearities, or develop a set of simple models and select among them using the current workload phase and control setting. This work takes the latter approach for the following reasons. A single complex thermal model requires numerous assumptions about the relationship between temperature and workload, which in turn requires extensive manual validation effort for each chip design. The model designer must choose which workload metrics are most relevant to temperature by developing a power model and a thermal model. Fine-grained power models are difficult to verify because chips are not instrumented to measure power at the microarchitectural level. Modeling heat flow with thermal models require extensive floorplan knowledge and is strongly dependent on the chip-cooling apparatus. Each chip floorplan can differ dramatically in terms of spatial power distribution and temperature sensor layout, so this process must be repeated for each new design. Many previous works circumvent this complexity by using simplified empirical formulas for power consumption as a function of cycles-per-instruction (CPI), frequency, and voltage [5, 6]. Our experiments in Section 5.5 show that the over-generality of these formulas can introduce enough estimation error to offset the benefits of predictive control.

As stated in Section 5.1, we learn $\widehat{g}(\cdot,\cdot)$ in Equation 2.1 for each workload phase and DVFS setting. The values of a_{mn} and a_m^{idle} are learned by observing the response of idle cores while running thermally aggressive workloads on adjacent cores. By assuming that the value of $\widehat{g}(\cdot,\cdot)$ is zero for idle cores, the remaining model parameters are estimated using least-squares regression on the training data. Because these values reflect heat conductions through static physical material, we allow them to remain constant for all phases and settings. We estimate $\widehat{g}(\cdot,\cdot)$ for each phase and DVFS setting to approximate the training data observations $g_m[i]$ defined in Equation 5.1.

$$g_m[i] = T_m[i] - \sum_{n=1}^{N} a_{mn}T_n[i-1] - a_m^{idle} \tag{5.1}$$

A detailed description of our phase classification techniques is given in Section 5.3. At the highest level, the phase classifier takes per-core performance counter values and estimates the probability that a sample is in a particular phase. These metrics span a wide range of architectural behavior. The classifier input values are normalized to the number of core cycles in order to give consistent readings across all DVFS settings and utilization levels. While these inputs capture a wide range of complex workload behavior, many exhibit high correlations (e.g., cache read performance counters are correlated with the retired instructions performance counter). Thus, in order to reduce the number of classifier parameters, we use *principal component analysis* (PCA) as a method to reduce the input dimensionality. PCA transforms a number of correlated variables into a smaller number of uncorrelated variables, or *principal components*, while retaining most of the original information [54]. Before applying PCA, we subtract the mean values and normalize the inputs in order to account for the different units of measurement. The input to the phase classifier in then the first two principal components of the performance counter values.

5.3 Workload Phase Classification

It is well established that workloads execute in consistent and repetitive patterns [55, 103, 51, 15], and this fact is reflected in all manifestations of workload behavior (computational operations, memory operations, power, temperature etc.). These behaviors can change instantaneously and dramatically within a workload or across a mixture of workloads. Thus, the average behavior of an entire workload gives an incomplete picture. For instance, as a workload enters a new control path in its execution, it can go from being CPU-bound to being memory-bound, which has potentially large implications for control. Many state-of-the-art control techniques leverage these fine-grained transient variations by performing workload phase classification.

For control purposes, a *workload phase* is defined as a period of workload execution that exhibits a consistent control response that is distinguishable from other phases. Workload phase detection amounts to a classification problem in which phase probabilities/assignments are calculated as a function of the classifier inputs for a period of workload execution. The controller then uses the control model prediction associated with the likeliest phase or a weighted average of the model predictions to perform control.

The phase classifier input vector can be defined in a number of ways, and does not need to have the same inputs as the control models. A number of previous works use traversal counts on basic block vectors (BBV), which define regions of code with a single entry and exit point [103, 51]. Other works use metrics derived from performance counters such as IPC and cache-misses per cycle [51, 22]. We define the phase classifier inputs using performance counters for three reasons. First, they are readily available on most commercial processors, making our approaches widely applicable. Second, they can be measured in real-time with minimal overhead. And finally, unlike BBVs, they capture workload behavior without prior knowledge about the workload structure.

We explore two techniques for learning phase definitions given a set of training data. The first technique, K-means clustering, is an *unsupervised* classifier that that we explored in our previous work [22]. The K-means algorithm defines a set of K phases by performing clustering on the classifier inputs. The main drawback of using the unsupervised K-means approach is that there is no guarantee that the control response is consistent within each phase. The tasks of defining workload phases and learning the associated control models are mutually dependent. The subset of the training data that is associated with each phase affects the model parameter estimates, while the choice of model parameters determines the phase boundaries. We develop a more sophisticated approach that handles this mutual dependence by defining *phase probabilities* for each sample as a function of the phase inputs. We introduce an expectation maximization (EM) algorithm that iteratively alternates between calculating phase probabilities and model estimation such that the log-likelihood of the training data is guaranteed to increase. This approach is initialized with and iteratively improves upon the K-means clustering phase assignments.

5.3.1 Phase Classification Using K-Means Clustering

A standard technique for defining workload phases in the literature is K-means clustering [103]. In our previous work [22], we extend workload phase classification for the purposes of thermal control by associating each phase with a control model predicting temperature as a function of the DVFS setting. The inputs to the K-means classifier are the first two principal components of the performance counter metrics. During runtime, the classifier chooses the likeliest control model by associating the incoming performance counter measurements with one of K predefined clusters. The associated thermal model is then used to guide control decisions. The intuition behind this approach is that workload intervals showing similar performance counter values are likely to have similar control responses. Likewise, workload intervals with drastically different performance counter values are likely to have distinct control responses.

The optimal cluster definitions are defined offline using a set of training data gathered across a range of workload behaviors. Each of the M samples in our training data is associated with a d-dimensional vector $\mathbf{x_m}[i]$ of performance counters, which are the phase classifier input for core m at time instant i. Each phase k is defined with a d-dimensional centroid, and each sample $\mathbf{x_m}[i]$ belongs in the phase associated with the closest centroid (Euclidean distance). The K-means clustering algorithm iteratively seeks a set of K centroids that minimizes the average distance between each point and the closest centroid. Determining the optimal centroid locations is an NP-hard problem; therefore, all algorithm implementations are heuristic and are not guaranteed to converge to a global optimum. The K-means algorithm alternates between an assignment step where each observation is assigned to the closest cluster centroid, and an update step where the cluster centroids are updated based on the latest cluster assignments [66]. An average value is calculated for $g_m[i]$ for each phase and DVFS setting combination. At runtime, we select the average value corresponding to the current phase and future setting to be the estimate for $\widehat{g}(\cdot, \cdot)$.

5.3.2 Phase Classification Using Multinomial Logistic Regression (MLR)

This section develops a supervised expectation-maximization (EM) template that improves upon the K-means approach. The tasks of defining workload phases and learning the associated control models are mutually dependent. The subset of the training data that is associated with each phase affects the model parameter estimates, while the choice of model parameters determines the phase boundaries. The unsupervised K-means approach estimates the phase assignments within the training data by clustering the classifier inputs, and then calculating the average value of $g_m[i]$ for each setting and phase. Defining the workload phases independently of the control models can lead to suboptimal prediction accuracy, as there is no guarantee that each phase will exhibit consistent thermal behavior. To handle the mutual dependence, we develop a more sophisticated expectation-maximization (EM) approach that is initialized with the K-means output and iteratively improves the phase definitions and model estimates. Given a set of training data and a set of assumptions about the control model, including the number of distinct models to be used, the algorithm is guaranteed to increase the log-likelihood of the training data with each iteration.

The main drawback of K-means classification is that its learning objective is not a function of the desired phase assignments. It merely performs clustering as a function of the inputs. Thus, we redefine the phase classifier with a multinomial logistic function in Equation 5.2, which maps a vector of K continuous values to a set of K discrete probabilities that sum to 1. Each vector value then parameterized with the dot product of a d-dimensional set of logistic model weights $\mathbf{w_k}$, which are learned as a function of the desired phase probabilities, and the phase classifier inputs $\mathbf{x_m}[i]$.

$$\pi_{mki} = \frac{\exp\left(\mathbf{w_k} \cdot \mathbf{x_m}[i]\right)}{\sum_{k'=1}^{K} \exp\left(\mathbf{w_{k'}} \cdot \mathbf{x_m}[i]\right)} \tag{5.2}$$

The variable π_{mki} denotes the probability that sample i for core m is in phase k given input $\mathbf{x_m}[i]$. In Equation 5.3, we define ϕ_{mki} to be the likelihood of observing output $g_m[i]$ with setting s in phase k given the expected value μ_{sk} and error variance σ_{sk}^2, both of which are model parameters that must be estimated.

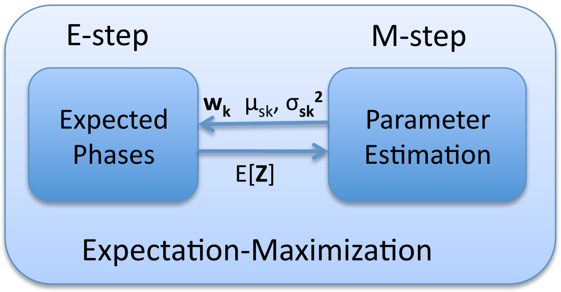

Figure 5.2: Illustration of the EM approach.

$$\phi_{mki} = \mathcal{N}\Big(g_m[i] \;\mid\; \mu_{sk},\;\; \sigma^2_{sk}\Big) \tag{5.3}$$

We also introduce a binary variable $z_{mki} \in \{0, 1\}$, which indicates whether sample i for core m is in phase k. If the "true" phase assignments were known, then the logistic and control model weights could be learned by directly maximizing the training data log-likelihood,

$$L(\mathbf{Z}) = \log \prod_{m=1}^{N} \prod_{k=1}^{K} \prod_{i=1}^{M} \Big(\pi_{mki} \;\; \phi_{mki}\Big)^{z_{mki}}, \tag{5.4}$$

where $\mathbf{Z}$ is the concatenation of z_{mki} for all values of m, k, and i. However, with the phase assignments unknown, we make an initial guess for $E[\mathbf{Z}]$ using the K-means output. The logistic weights $\mathbf{w_k}$ and values for μ_{sk} and σ_{sk} are then estimated such that $L(E[\mathbf{Z}])$ is maximized. The value of $E[\mathbf{Z}]$ can then be recomputed given the latest model estimates. With each iteration, the observed data log-likelihood is *guaranteed* to increase, improving the model to fit to the training data. The algorithm iterates between the maximization and expectation step in this manner, depicted in Figure 5.2, until $L(\mathbf{Z})$ converges within a predefined tolerance. Each EM iteration is summarized with the following:

Maximization Step

$$\Big[\mathbf{w_k},\ \ \mu_{sk},\ \ \sigma_{sk}^2\Big]_{\forall s,k} = \operatorname{argmax} L\Big(E[\mathbf{Z}]\Big) \tag{5.5}$$

Expectation Step

$$E\Big[z_{mki}\Big] = \frac{\pi_{mki}\ \ \phi_{mki}}{\sum\limits_{k'=1}^{K} \pi_{mk'i}\ \ \phi_{mk'i}} \tag{5.6}$$

During runtime, the workload thermal contribution $\widehat{g}(\cdot,\cdot)$ is calculated as a weighted sum of average observed values for $g_m[i]$ in the training data, or

$$\widehat{g}\Big(s,\ \ \mathbf{x_m}[i]\Big) = \sum_{k=1}^{K} \pi_{mki}\ \ \mu_{sk}. \tag{5.7}$$

By defining phase probabilities instead of hard assignments, we are naturally able to interpolate between thermal model estimates when the workload phase is uncertain. For example, if each of the K workload phases determined to be equiprobable for phase input $\mathbf{x_m}[i]$, then the average value across all phases will be used for prediction. This is a stark improvement over the hard K-means, which does not allow for uncertainty in phase detection, and may produce unstable predictions if a workload straddles a phase boundary. The interface between our offline and runtime approaches is illustrated in Figure 5.3. The phase classifier input means, magnitudes, and principal component loadings, as well as the phase definitions and thermal models learned using EM, are conveyed to the runtime DTM controller using lookup tables.

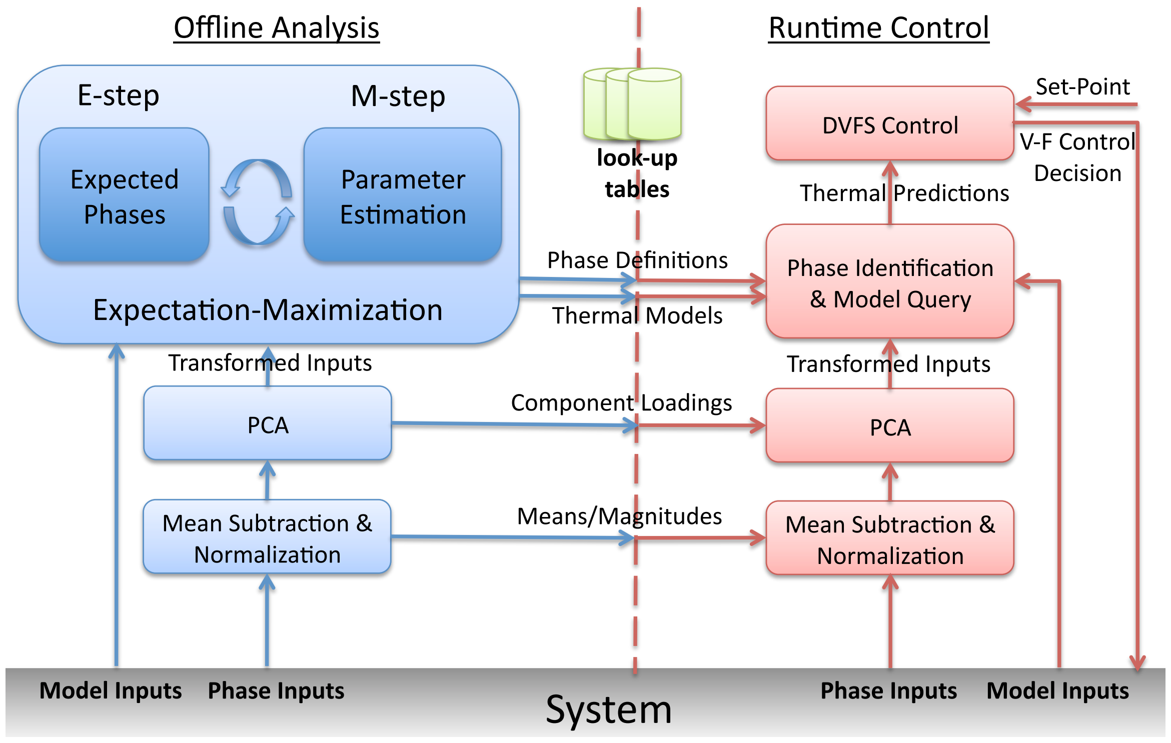

Figure 5.3: Lookup table interface between our offline and runtime DTM methodologies.

5.4 Runtime Control

Periodically during runtime, the DVFS control unit makes a control decision as function of the projected temperature. Thermal predictions are based on performance counter and temperature data accumulated over the previous control interval. Before phase identification, the phase inputs are transformed as described in Section 5.2. The input averages, normalization constants, and principal component loadings are conveyed in the lookup tables. Once transformed, the first two principal components are used in conjunction with the MLR weights for phase identification according to Equation 5.2, which estimates the probability of the being in a particular phase. The thermal model associated with the likeliest workload phase is subsequently used to make thermal predictions as a function of each potential DVFS control decision.

The DVFS control technique uses these thermal predictions in order to maximize performance within a given thermal threshold. The thermal model associated with the current workload phase k is queried for predictions as a function of each DVFS setting using the state-space model in Equation 2.1 and the workload thermal contribution $\widehat{g}(\cdot,\cdot)$ defined in Equation 5.7. The setting with the maximum frequency such that the maximum projected core temperature is below the temperature set-point is selected by the DVFS controller. Given frequency f which is a function of the current setting DVFS setting s, this can be expressed as

$$s_{sel} = \operatorname*{argmax}_{s} f(s) : \max_{m}(T_m[i+1]) < T_{set}. \tag{5.8}$$

Because all model learning is performed offline, online prediction incurs negligible overhead. Each control decision incurs 2-4 ms of overhead, which represents a performance overhead of $0.2 - 0.4\%$ assuming a control activation period of 1 seconds. We choose a 1 second control activation period to match the minimum sampling period of the thermal sensing hardware on our test platform. Because our model is not a linear function of processor frequency, it is incompatible with the quadratic model predictive controller (MPC) techniques used in previous works on thermal control. However, our results in Section 5.5 show by combining the superior modeling accuracy of our phase-aware approaches with a soft/hard threshold control scheme, we are able to significantly reduce hard limit violations and improve control relative to the MPC approach.

5.5 Experiment Results

This section provides the details of the experimental setup, the results of our workload phase thermal characterization techniques, and the results of our control experiments. Our experimental setup consists of the following:

- All collection and control experiments are performed on an Intel Core i7 940 45nm quad-core processor, running the 2.6.10.8 Linux kernel OS.
- Performance counter data are collected using the `pfmon` (version 3.9) utility. We poll performance counters for each core at 1 second intervals. The phase inputs $\mathbf{x_{mi}}$ are identified in Figure 5.6.
- Each core on the Core-i7 processor is equipped with a digital thermal sensor, measuring the maximum junction temperature. The pfmon tool is interfaced with the Linux `lm-sensors` library to report these per-core temperatures at 1 second intervals.
- The Core-i7 processor V-F settings are manipulated with the `cpufreq` utility, and the available frequency settings are: $\{$1.60 GHz, 1.73 GHz, 1.87 GHz, 2.00 GHz, 2.13 GHz, 2.27 GHz, 2.40 GHz, 2.53 GHz, 2.67 GHz$\}$. The voltages associated with each frequency are set automatically in hardware. As is the case with most commercial multi-core processors, our test platform lacks per-core frequency domains. Each DVFS control decision is applied globally to all cores.
- To implement data collection and runtime control, we interface our data measurement and control apparatus to a MATLAB module compiled as a C-shared library. This module is configured to read lookup tables generated offline, buffer incoming performance counter and temperature data, and perform control every 1 second. We choose 1 second as our control activation period because this is the minimum update time for the thermal

sensing hardware on our test platform. The runtime overhead for each activation of the control algorithm during runtime is in the range of 2-4 ms. While this runtime overhead is 0.2-0.4 % of the control activation period, it is worth noting that because our control algorithm is implemented in MATLAB scripting and is run using the Java Virtual Machine (JVM), this overhead could be reduced even further if implemented with optimized C code. In general, we observe that for a particular workload, the duration of each workload phase is on the order of 10 seconds. The storage requirements of our lookup tables are minimal and include the following: 36 μ_{sk} and σ_{sk} values corresponding to every frequency and workload phase combination, 20 values corresponding to a_{mn} and a_m^{idle} for each core, and 12 values corresponding to the MLR classifier weights $\mathbf{x_{mi}}$ for each workload phase and principal component plus a constant term.

5.5.1 Offline Characterization

For offline characterization, we build a set of training data using thermally "interesting" workloads from the SPEC CPU2006 benchmark suite (`astar`, `bzip2`, `gcc`, `calculix`, `dealII`, and `tonto`). While the majority of the SPEC CPU2006 workloads exhibit relatively flat thermal profiles during execution, these workloads demonstrate significant temperature fluctuations even with a static DVFS setting. The dynamic thermal behavior of these workloads make them ideal for training data. Each workload is executed for 15 minutes with 1 to 4 instances in order to expose the thermal response to changes in processor utilization. At the same time, we systematically throttle the DVFS setting every 5 seconds such that every transition is exercised. Each training data sample includes per-core values for the 12 performance counters listed in Figure 5.6, as well as per-core temperature measurements and the current processor frequency. By applying PCA, we are able to reduce this set of 12 raw phase classifier inputs to 2 principal components while retaining the majority of observed variance.

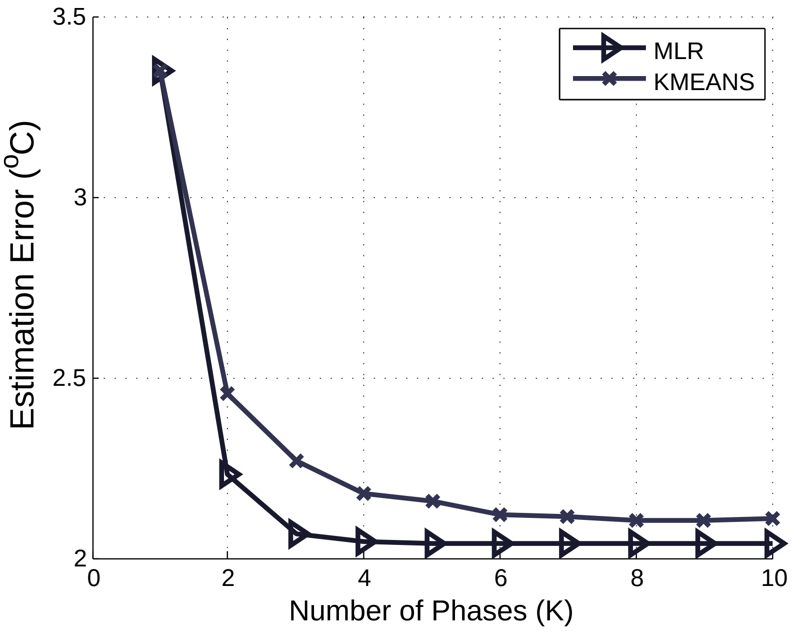

Figure 5.4: The estimation accuracy for the proposed K-means and MLR approaches as a function of the number of phases K. Reported accuracy is 3σ, where σ is the error standard deviation.

The appropriate value for the number of phases K is determined experimentally using our training data. To prevent "overfitting" , we randomly select 10000 samples from our training data set of 21480 points to train our thermal models. We then evaluate the prediction accuracy on the remaining data points. By incrementally increasing K, we are able to determine the optimal number of phases beyond which there is no decrease in thermal prediction accuracy. The results in Figure 5.4 show that 4 workload phases accurately capture the salient workload dependencies across all DVFS settings for both K-means and MLR phase classification. The reported accuracy is reported as 3σ, where σ is the error standard deviation. Given that the errors are normally distributed, this metric indicates the error magnitude that $< 0.1\%$ of the training samples violate. Figure 5.4 shows that the MLR approach yields superior estimation accuracy to K-means for all values of K.

Figure 5.5 shows the phase boundaries that results from the 4-phase MLR approach in principal component space. The black lines indicate the projection of each performance

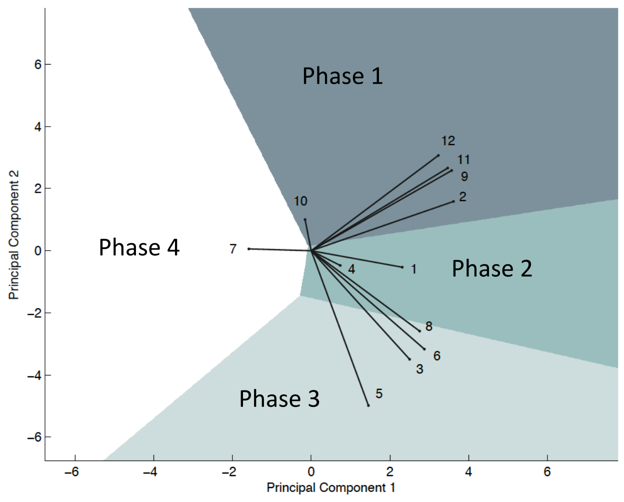

Figure 5.5: Phase boundaries in principal component space. The number labels correspond to the performance counters listed in Figure 5.6.

	Description
1	Branch prediction misses
2	Conditional branch instructions
3	Resource stalls
4	Floating-point operations
5	L3-cache misses
6	L2-cache misses
7	Load locks
8	L1-data-cache misses
9	L1-data-cache reads
10	L1-instruction-cache misses
11	L1-instruction-cache reads
12	Micro-operations retired

Figure 5.6: List of 12 performance counters used for phase classification.

counter metric in Figure 5.6 onto the first 2 principal components. Figure 5.7 illustrates the per-core steady-state magnitudes associated with each phase on our quad-core test platform. It is assumed for illustration purposes that all cores are in the same workload phase. Comparing Figures 5.5 and 5.7 reveals that phase 1, which is the most thermally aggressive phase, corresponds to high instruction throughput with low data cache miss rates. As the the memory activity (memory-boundedness) increases in phases 2 and 3, the chip temperature becomes less sensitive to processor frequency. Phase 4 represents the idle-core temperatures in which all performance counter metrics are low.

We break down the estimation error for each training workload and utilization level in Figures 5.8 and 5.9 respectively. We compare our proposed phase-aware MLR and K-means modeling techniques to three alternative approaches: linear regression (LR), 1-Phase, and the MPC power modeling technique in [5, 6]. In the LR approach, we model the power consumption in the state-space model as a weighted combination of the the 12 performance counter metrics in Figure 5.6, where a set of weights are learned for each

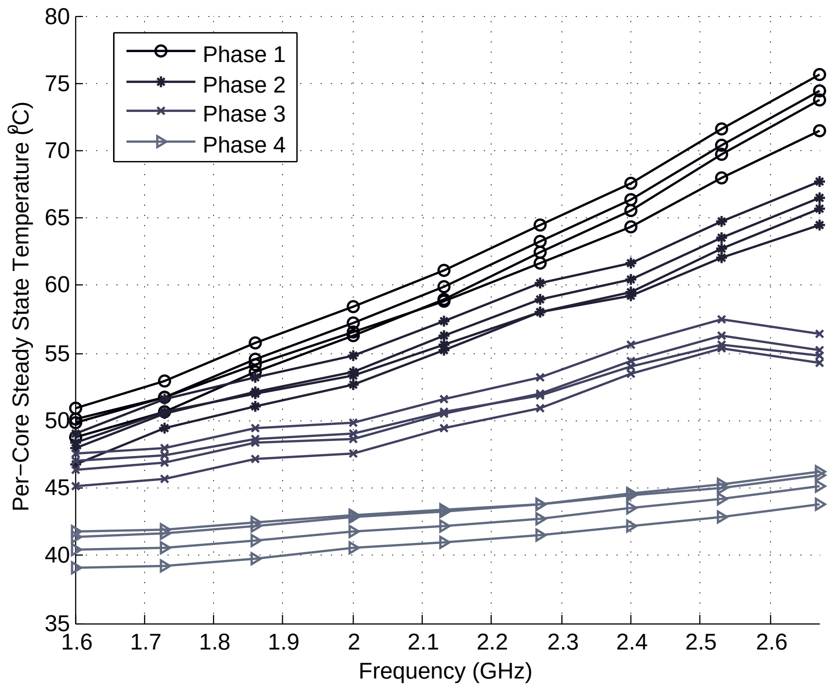

Figure 5.7: Per-core steady-state temperatures for each phase as a function of processor frequency.

DVFS setting. Because the workload phase approach implicitly captures nonlinearities, we are able to show superior prediction accuracy relative to a single linear regression model. The 1-Phase approach is our proposed MLR approach but with a single workload phase. This comparison quantifies the benefit of using workload phases. The MPC approach models the power consumption according to Equation 2.2, which is a verified empirical formula for power estimation that appears in previous works [5, 6]. Using CPU power measurements in our training data, we estimate the parameters of this model using nonlinear regression. Despite being able to estimate the total power consumption with an average absolute error of 0.62 W, this power model leads to significant estimation error when performing thermal prediction with the state-space model. In addition to giving superior temperature accuracy, our phase-aware approaches do not require intermediate power measurements in order to learn a thermal model. Our automated approach to linearizing workload dependence can be performed on any system equipped with ther-

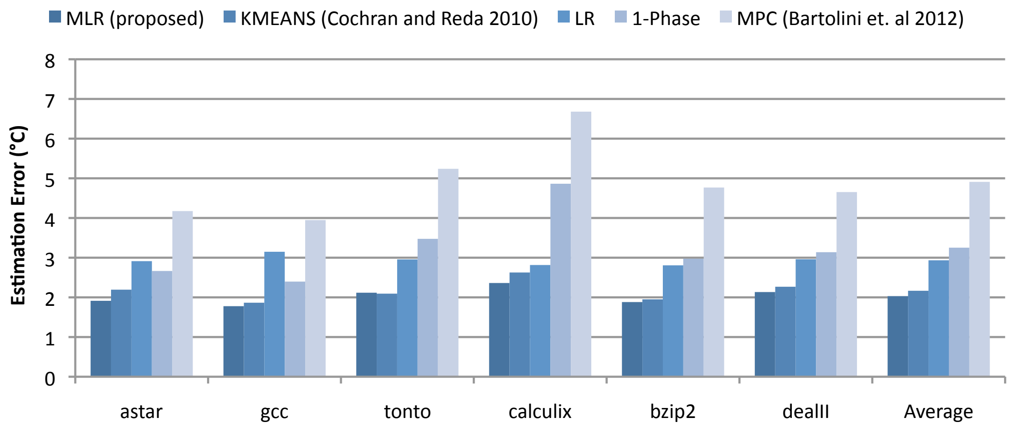

Figure 5.8: Estimation error (3σ) breakdown for each training workload.

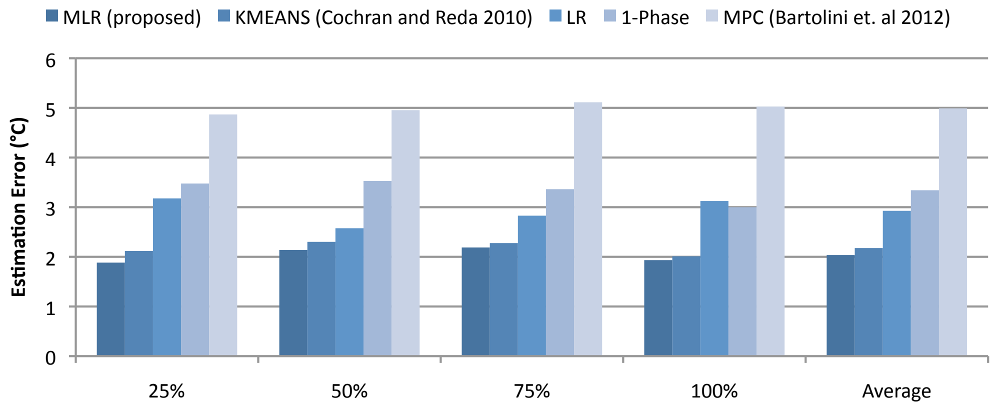

Figure 5.9: Estimation error (3σ) as a function of processor utilization.

mal sensors and performance counters. The results show accuracy improvement for the proposed approaches across the board, with a maximum 59% improvement in prediction accuracy relative to the MPC method.

5.5.2 Runtime Control

To verify that prediction accuracy translates to improved thermal control, we perform a control experiment in which we integrate the proposed technique into the DVFS control of a Core-i7 quad-core processor. We quantify our approach's ability to maximize performance within a temperature constraint for a variety of heterogenous workload combinations. We evaluate each technique use a sequence of 15 combinations of between 1 and 4 workloads selected from the entire suite of SPEC CPU2006 workloads. In doing this, we evaluate each modeling technique for a range of workload behaviors and utilization levels. We compare our proposed MLR approach and previous K-means approach [22] to the following alternatives:

- 1-Phase: This is the proposed approach but with a single workload phase. This corresponds to a single estimate for $\widehat{g}(\cdot, \cdot)$ for each DVFS setting. Comparison to this approach quantifies the benefit of adding workload phases.

- SPRT: We use the set of models learned using MLR but use the SPRT phase switching technique [25, 23].

- MPC: In this approach, we use the power model described in Section 5.5.1 [5, 6]. Because this model is linear with frequency, we can incorporate it into a quadratic model prediction control (MPC) objective in place of the objective described in Section 5.4. With each control interval, the MPC controller predicts a sequence of DVFS settings that minimizes a quadratic cost function while maintaing the temperature within the thermal set-point. The cost function is the squared deviation from the hard threshold with a prediction horizon of 3 control intervals (3 seconds).

- PI: We compare all modeling techniques to a simple proportional-integral PI feedback technique inspired by the reactive techniques used in commercial processors [7]. We calibrate the proportional and integral coefficients to the temperature-frequency gain observed for full processor utilization (K_p = 0.12 GHz/$^{\circ}C$, K_i = 0.04 GHz/$^{\circ}C$). The integral term is the sum of the soft-threshold deviations from the previous 3 control intervals.

	proposed			previous		
	MLR	KMEANS	1-Phase	SPRT	MPC	PI
Violations	2	5	8	76	2	33
Average magnitude (°C)	1.00	1.00	1.25	1.62	2.00	2.12
Temperature variance (°C^2)	1.57	1.24	2.42	7.46	4.42	5.37
Instruction throughput (10^9 / sec)	6.06	6.00	5.88	5.89	5.73	5.83

Table 5.1: Comparison of thermal violations and performance of proposed phase-aware techniques to previous techniques.

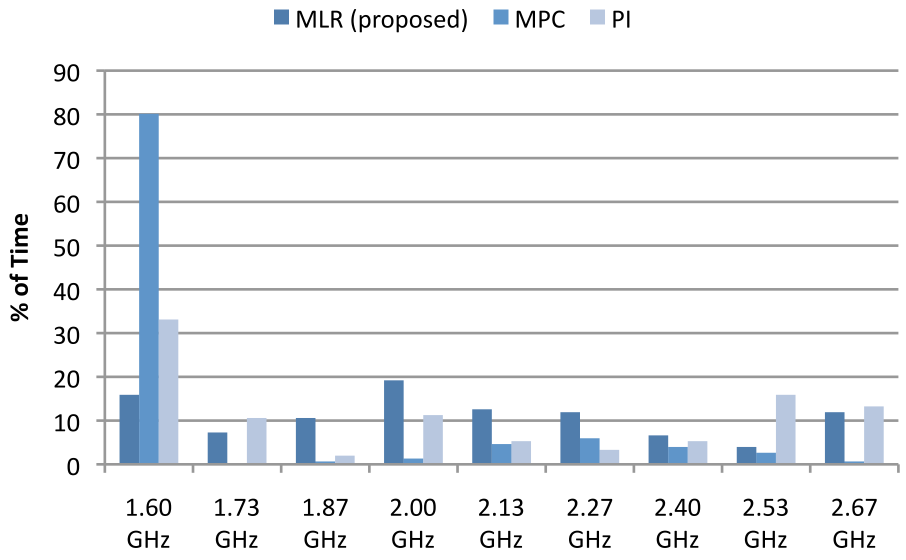

Figure 5.10: Frequency histogram comparison for MLR, MPC, and PI controllers.

For each approach, we use an aggressive hard thermal threshold of 55$°C$ and we repeat each experiment for integer soft threshold values within the range of 50-55$°C$. In Table 5.1, we report results for the proposed MLR method with a soft threshold of $50°C$. In order to compare to MLR, for each alternate proposed and previous approach we report results for the soft threshold that produces the smallest number of thermal violations greater than or equal to the number of violations produced by MLR at 50$°C$. The results show that the superior workload dependence modeling for the MLR approach allows it to achieve fewer thermal violations *and* higher instruction for time intervals in which the temperature constraint is met. We also evaluate the temperature standard deviation and show that the proposed MLR approach significantly reduce thermal oscillations and asso-

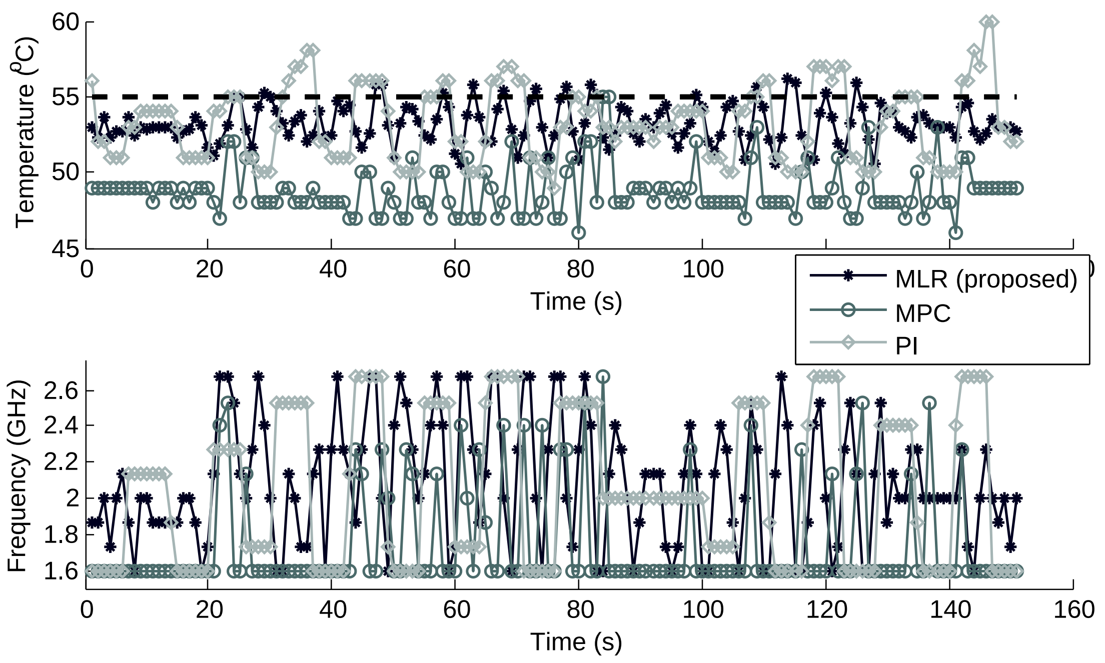

Figure 5.11: Comparison of thermal traces for MLR, MPC, and PI controller for 150 second interval.

ciated mechanical stresses, which translates into improved reliability. The MPC control objective, which uses the CPI power model, overestimates the temperature response to changes in DVFS setting. The decrease in prediction accuracy is not enough to offset the benefits of the quadratic MPC objective, and leads to overly cautious DVFS control. The MLR approach yields a significant 5.8% improvement in instruction throughput. The PI controller, while slightly less pessimistic, is calibrated to the case of maximum utilization. As a a result, it too overestimates the temperature response and yields suboptimal frequency selection during periods of low utilization. The MLR approach improves the performance 3.9% over the PI controller. While MLR and K-means produce comparable results, the improved accuracy of the MLR approach yields slightly fewer violations and higher performance.

In Figure 5.10, we show the a comparison of frequency selections for the MLR, MPC, and PI approaches. The results show the the overly sensitive MPC and PI approaches are forced to use the lowest frequency for a larger percentage of the time, whereas the

MLR method is able to select higher frequencies while still meeting the thermal constraint. We illustrate this further with the temperature traces in Figure 5.11. The MPC controller consistently maintains a lower average temperature, while the PI controller produces larger thermal violations but lower average temperature.

5.6 Summary

In this chapter, we devise and verify a thermal prediction strategy for use in DVFS thermal control. We capture complex workload dependencies and nonlinearities using a set of simple thermal models associated with workload phases. Our approach leverages the fact that workloads within a local window of performance counter values have similar thermal contributions. We develop classifiers that partition the space of workload behaviors into phases, and we calculate a distinct value of $\widehat{g}(\cdot, \cdot)$ for each DVFS setting within each partition. We derive a novel application of the expectation-maximization (EM) algorithm using multinomial logistic regression (MLR) to simultaneously learn workload phases and the associated thermal models. By estimating average power consumption independently for each DVFS setting and workload phase, we implicitly capture nonlinearities. The overall model complexity is controlled by the number of phases K, the optimal value for which is determined experimentally. We use training data gathered from an extensive experimental setup on a quad-core system to estimate phases and thermal models. We demonstrate that our workload phase approach provides superior prediction accuracy compared the models used in previous works. We then show that this improved prediction accuracy translates to superior DVFS thermal control by reducing thermal violations and increasing performance. In comparison to state-of-the-art model predictive control (MPC) techniques in previous works on thermal control, we demonstrate a 12.4% improvement in instruction throughput with a negligible increase in the number of thermal violations. In comparison to simple proportional-integral (PI) feedback control techniques, we improve instruction throughput by 8.4% with an 80% reduction in the number of thermal violations.

Chapter 6

Conclusions

In this thesis, we sought to address two primary questions for adaptive workload-sensitive DPM and DTM.

1. How can the processor hardware supply accurate power and thermal information to DPM and DTM controllers using a limited number of sensors?
2. How can a device perform adaptive workload-sensitive DPM and DTM using the measurements available at runtime?

We now assess the contributions of this thesis in answering these questions and discuss future research extensions.

In Chapter 3, we develop infrastructure for measuring power consumption and die temperatures of real processors with a state-of-the-art thermal infrared camera. Based on the characterization results from the camera, we propose a formulation for thermal sensor allocation to minimize the thermal tracking error during runtime. We prove that our formulation leads to a NP-hard problem and accordingly we propose a heuristic solution method that is composed of constructive and iterative phases. The experimental results show that our method significantly improves upon methods in the literature. To circumvent limitations on thermal sensor placement, we proposed two soft sensing techniques that combine the measurements of hard sensors to estimate the temperatures at the ideal locations. Our first technique leverages *a priori* design-time characterization data to seek customized weighted combinations to estimate the temperatures at any desired locations. Our second technique uses spectral signal reconstruction techniques based on Fourier analysis for interpolating die temperatures from a limited number of thermal sensors. This soft-sensing technique is designed for cases in which there is no *a priori* thermal characterization either through simulation or infrared camera. We show that both soft-sensing techniques significantly improve thermal estimation accuracy. In the future, the techniques laid out in this chapter will be used to characterize and deal with the additional challenges of many-core architectures.

In Chapter 4, we introduce thread reduction for multithreaded workloads as a means of meeting low power budgets on a single server node, thus increasing power allocation flexibility for datacenters composed of many server nodes. We then propose *thread packing*, in which multi-threaded workloads are packed onto a variable number of active cores, as a flexible proxy for thread reduction. We then devise a novel DPM method, Pack & Cap, which makes optimal DVFS and thread packing control decisions such that performance is maximized within a power budget. We illuminate power and performance tradeoffs between thread packing and DVFS and develop heuristics for navigating Pareto efficient control settings. We implement Pack & Cap with several candidate power models, including a multi-gain feedback controller as well as open-loop and closed-loop linear regression models. We compare these techniques to a baseline feedback technique and demonstrate up to 10% reduction in average runtime and up to 53% improvement in average power cap accuracy. We also compare to our previous work in [18], which uses a multinomial logistic regression (MLR) classifier, and show up to 30% runtime improvement with similar power cap accuracy. In the future research, this work on individual server node power capping can be integrated into a group power capping scheme in which sets of nodes are managed together under a common power constraint. In order for this work to be widely applicable, it will need to be extended to heterogenous architectures as well. And finally, this methodology can be extended to differentiate between critical and non-critical threads and compared to previous works which do the same [14, 8].

In Chapter 5, we introduce a novel technique for estimating the temperature control response that uses workload phases. We use classification techniques from machine learning to classify workload execution intervals into phases as a function of performance counter measurements. This method addresses the inherent nonlinearities of workload-sensitive temperature modeling on multi-core systems, and we demonstrate improved accuracy over a linear regression model. Although this model can be applied to many of the control objectives in the literature, we demonstrate this technique in the case of proactive DVFS thermal control. We compare to state-of-the-art model predictive control (MPC) control techniques in the literature and show 5.8% improvement in instruction throughput with the same number of thermal violations. When we compare our thermal phase classification techniques to the sequential-probability-ratio-tests (SPRT) used for switching thermal models in previous works, we demonstrate a 2.9% improvement in performance and an 97% reduction in thermal violations. In comparison to optimally tuned proportional-integral (PI) feedback control techniques, we improvement instruction throughput by 3.9% with an 94% reduction in the number of thermal violations. In future extensions, the thermal modeling approach using the discrete state-space model and workload phases should be integrated into more complex control objectives in order to further verify its utility. First and foremost, it can be integrated with the work in Chapter 4 by adding a thermal constraint to the Pack & Cap objective.

Bibliography

[1] Advanced Configuration and Power Interface Specification. http://ita.ee.lbl.gov/html/traces.html.

[2] E Alpaydin. *Alpaydin: Introduction to Machine Learning*. Cover.

[3] R. I. Bahar and S. Manne. Power and Energy Reduction Via Pipeline Balancing. In *Proceedings of the International Symposium on Computer Architecture*, pages 218–229, 2001.

[4] L. A. Barroso. *The Datacenter as a Computer*. Morgan Claypool, 2009.

[5] A. Bartolini, M. Cacciari, A. Tilli, and L. Benini. A Distributed and Self-Calibrating Model-Predictive Controller for Energy and Thermal Management of High-Performance Multicores. In *Proceedings of Design, Automation and Test in Europe Conference*, pages 1–6, 2011.

[6] A. Bartolini, M. Cacciari, A. Tilli, and L. Benini. Thermal and Energy Management of High-Performance Multicores: Distributed and Self-Calibrating Model-Predictive Controller. *IEEE Transactions on Parallel Distributed Systems*, 2012.

[7] M. Berktold and T. Tian. CPU Monitoring With DTS/PECI. http://download.intel.com/design/intarch/papers/322683.pdf.

[8] A. Bhattacharjee. Thread Criticality Predictors for Dynamic Performance, Power, and Resource Management in Chip Multiprocessors. In *Proceedings of International Symposium on Computer Architecture*, pages 290–301, 2009.

[9] C. Bienia. *Benchmarking Modern Multiprocessors*. PhD thesis, 2011.

[10] S. Borkar. Thousand core chips - a technology perspective. In *Proceedings of the Design Automation Conference*, pages 746 – 749, 2007.

[11] D. Brooks, R. Dick, R. Joseph, and L. Shang. Power, Thermal, and Reliability Modeling in Nanometer-Scale Microprocessors. *IEEE Micro*, 27(3):49 – 62, 2007.

[12] D. Brooks and M. Martonosi. Dynamic Thermal Management for High-Performance Microprocessors. In *Proceedings of High Performance Computer Architecture*, pages 171–182, 2001.

[13] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In *Proceedings of the International Symposium on Computer Architecture*, pages 83–94, 2000.

[14] J. M. Cebrian, J. L. Aragón, and S. Kaxiras. Power Token Balancing: Adapting CMPs to Power Constraints for Parallel Multithreaded Workloads. In *Proceedings of Parallel & Distributed Processing Symposium*, pages 431–442, 2011.

[15] C. B. Cho and T. Li. Complexity-Based Program Phase Analysis and Classification. In *Proceedings of Parallel Architectures and Compilation Techniques*, pages 105–113, 2006.

[16] A Çivril. On Selecting a Maximum Volume Sub-Matrix of a Matrix and Related Problems. *Theoretical Computer Science*, 2009.

[17] R. Cochran, C. Hankendi, A. K. Coskun, and S. Reda. Identifying the Optimal Energy-Efficient Operating Points of Parallel Workloads. In *Proceedings of the International Conference on Computer-Aided Design*, pages 608–615, 2011.

[18] R. Cochran, C. Hankendi, A. K. Coskun, and S. Reda. Pack & Cap: Adaptive DVFS and Thread Packing Under Power Caps. In *Proceedings of the International Symposium on Microarchitecture*, pages 175–185, 2011.

[19] R. Cochran, A. Nowroz, and S. Reda. Post-Silicon Power Characterization Using Thermal Infrared Emissions. In *International Symposium on Low Power Electronics and Design*, pages 331–336, 2010.

[20] R. Cochran and S. Reda. Thermal Prediction and Adaptive Control Through Workload Phase Detection. *Revision currently under review for ACM Transactions on Design Automation of Electronic Systems, 2012.*

[21] R. Cochran and S. Reda. Spectral Techniques for High-Resolution Thermal Characterization with Limited Sensor Data. In *Proceedings of the Design Automation Conference*, pages 478–483, 2009.

[22] R. Cochran and S. Reda. Consistent Runtime Thermal Prediction and Control Through Workload Phase Detection. In *Proceedings of Design Automation Conference*, pages 62–67, 2010.

[23] A. Coskun, T. Rosing, and K. Gross. Utilizing Predictors for Efficient Thermal Management in Multiprocessor SoCs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(10):1503 – 1516, 2009.

[24] A. Coskun, T. Rosing, K. Whisnant, and K. Gross. Temperature-Aware MPSoC Scheduling for Reducing Hot Spots and Gradients. In *Proceedings of the Asia and South Pacific Design Automation Conference*, pages 49–54, 2008.

[25] A. K. Coskun, T. Rosing, and K. Gross. Proactive Temperature Balancing for Low Cost Thermal Management in MPSoCs. *Proceedings of the International Conference on Computer-Aided Design*, 2008.

[26] A. K. Coskun, T. S. Rosing, and K. C. Gross. Proactive Temperature Management in MPSoCs. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 165–170, 2008.

[27] G. Dhiman. Dynamic Voltage Frequency Scaling for Multi-Tasking Systems Using Online Learning. In *Proceedings of International Symposium on Low-Power Electronic Design*, pages 207–212, 2007.

[28] J. Donald and M. Martonosi. Techniques for Multicore Thermal Management: Classification and New Exploration. In *Proceedings of the International Symposium on Computer Architecture*, pages 78–88, 2006.

[29] E. Elnozahy, M. Kistler, and R. Rajamony. Energy-Efficient Server Clusters. In *Power-Aware Computer Systems*, volume 2325 of *Lecture Notes in Computer Science*, Berlin, Heidelberg, April 2003. Springer Berlin Heidelberg.

[30] S. Eyerman and L. Eeckhout. A Counter Architecture for Online DVFS Profitability Estimation. *IEEE Transactions on Computers*, 59(11):1576–1583.

[31] X. Fan, W. D. Weber, and L. A. Barroso. Power Provisioning for a Warehouse-Sized Computer. In *Proceedings of the International Symposium on Computer Architecture*, pages 13–23, 2007.

[32] D. Filani, J. He, S. Gao, et al. Dynamic Data Center Power Management: Trends, Issues, and Solutions. *Intel Technology Journal*, page 59, February 2008.

[33] M. Floyd, M. Allen-Ware, K. Rajamani, B. Brock, C. Lefurgy, A. J. Drake, L. Pesantez, T. Gloekler, J. A. Tierno, P. Bose, and A. Buyuktosunoglu. Introducing

the Adaptive Energy Management Features of the Power7 Chip. *Micro, IEEE*, 31(2):60–75, 2011.

[34] A. Gandhi, M. Harchol-Balter, and R. Das. Optimal Power Allocation in Server Farms. In *Proceedings of SIGMETRICS*, pages 157–168, 2009.

[35] A. Gandhi, M. Harchol-Balter, R. Das, and J. O. Kephart. Power Capping Via Forced Idleness. *Carnegie Mellon University Research Showcase*.

[36] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, first edition, 1979.

[37] H. Hamann, A. Weger, J. Lacey, Z. Hu, and P. Bose. Hotspot-Limited Microprocessors: Direct Temperature and Power Distribution Measurements. *IEEE Journal of Solid-State Circuits*, 42(1):56–65, 2007.

[38] H. F. Hamann, A. Weger, James A. L., Z. Hu, P. Bose, E. Cohen, and J. Wakil. Hotspot-Limited Microprocessors: Direct Temperature and Power Distribution Measurements. *IEEE Journal of Solid-State Circuits*, 42(1):56–65, 2007.

[39] G. Hamerly, B. Calder, T. Sherwood, and E. Perelman. Automatically Characterizing Large Scale Program Behavior. *Architectural Support for Programming Languages and Operating Systems*, 37, 2002.

[40] G. Hamerly, E. Perelman, and J. Lau. Simpoint 3.0: Faster and More Flexible Program Phase Analysis. *Journal of Instruction Level Parallelism*, pages 1–28, 2005.

[41] H. Hanson, S. W. Keckler, S. Ghiasi, K. Rajamani, F. Rawson, and J. Rubio. Thermal Response to DVFS: Analysis with an Intel Pentium M. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 219–224, 2007.

[42] J. Henning. SPEC CPU2000: Measuring CPU Performance in the New Millennium. *IEEE Computer*, 33(7):28–35, 2000.

[43] S. Heo, K. Barr, and K. Asanović. Reducing Power Density Through Activity Migration. In *International Symposium on Low Power Electronics and Design*, pages 217–222, 2003.

[44] S. Herbert and D. Marculescu. Analysis of Dynamic Voltage/Frequency Scaling in Chip-Multiprocessors. In *Proceedings of International Symposium on Low-Power Electronic Design*, pages 38–43, 2007.

[45] C. Hsieh, C. Wu, F. Jih, and T. Sun. Focal-Plane-Arrays and CMOS Readout Techniques of Infrared Imaging Systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(4):594–605, 1997.

[46] W. Huan, M. R. Stan, K. Sankaranarayanan, R. J. Ribando, and K. Skadron. Many-Core Design from a Thermal Perspective. In *Design Automation Conference*, pages 746–749, 2008.

[47] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan. HotSpot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Design. *IEEE Transactions onVery Large Scale Integration (VLSI) Systems*, 14(5):501 – 513, 2006.

[48] W. Huang, K. Skadron, S. Gurumurthi, R. J. Ribando, and Mircea R. Stan. Differentiating the Roles of IR Measurement and Simulation for Power and Temperature-Aware Design. In *Proceedings of the International Symposium on Performance Analysis of Systems and Software*, pages 1–10, 2009.

[49] W. Huang, M. Stan, K. Skadron, and K. Sankaranarayanan. Compact Thermal Modeling for Temperature-Aware Design. In *Proceedings of the Design Automation Conference*, pages 887–883, 2004.

[50] C. Isci, A. Buyuktosunoglu, C. Cher, P. Bose, and M. Martonosi. An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget. In *Proceedings of International Symposium on Microarchitecture*, pages 347–358, 2006.

[51] C. Isci, G. Contreras, and M. Martonosi. Live, Runtime Phase Monitoring and Prediction on Real Systems with Application to Dynamic Power Management. *Proceedings of the International Symposium on Microarchitecture*, pages 359–370, December 2006.

[52] C. Isci and M. Martonosi. Phase Characterization for Power: Evaluating Control-Flow-Based and Event-Counter-Based Techniques. In *Proceedings of International Symposium on High-Performance Computer Architecture*, pages 1–12, 2006.

[53] ITRS. International Technology Roadmap for Semiconductors. http://public.itrs.net, 2007.

[54] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, 6th edition, 2007.

[55] R. Joseph, M. Martonosi, and Z. Hu. Spectral Analysis for Characterizing Program Power and Performance. In *International Symposium Performance Analysis of Systems and Software*, pages 151–160, 2004.

[56] H. Jung and M. Pedram. A Stochastic Local Hot Spot Alerting Technique. In *Proceedings of the Asia and South Pacific Design Automation Conference*, pages 468 – 473, 2008.

[57] H. Jung, P. Rong, and M. Pedram. Stochastic Modeling of a Thermally-Managed Multi-core System. In *Proceedings of the Design Automation Conference*, pages 728 – 733, 2008.

[58] M. Kadin and S. Reda. Frequency and Voltage Planning for Multi-Core Processors Under Thermal Constraints. In *Proceedings of the International Conference on Computer Design*, pages 463–470, 2008.

[59] R. Kessler, E. McLellan, and D. Webb. The Alpha 21264 Microprocessor Architecture. In *Proceedings of International Conference on Computer Aided Design*, pages 90–95, 1998.

[60] O. Khan and S. Kundu. A framework for Predictive Dynamic Temperature Management of Microprocessor Systems. In *Proceedings of the International Conference on Computer-Aided Design*, pages 258–263, Jan 2008.

[61] J. Kim, S. Yoo, and C. Kyung. Program Phase-Aware Dynamic Voltage Scaling Under Variable Computational Workload and Memory Stall Environment. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(1):110–123, 2011.

[62] V. Kontorinis, A. Shayan, R. Kumar, and D. M. Tullsen. Reducing Peak Power with a Table-Driven Adaptive Processor Core. In *Proceedings of International Symposium on Microarchitecture*, pages 189–200, 2009.

[63] A. Kumar, Li S., Li-Shiuan P., and N. K. Jha. System-Level Dynamic Thermal Management for High-Performance Microprocessors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(1):96–108, 2008.

[64] A. Kumar, L. Shang, L.-S. Peh, and N. K. Jha. HybDTM: A Coordinated Hardware-Software Approach for Dynamic Thermal Management. In *Proceedings of the Design Automation Conference*, pages 548–553, 2006.

[65] P. Kumar and D. Atienza. Neural Network based On-Chip Thermal Simulator. In *Proceedings of International Symposium on Circuits and Systems*, pages 1599–1602, 2010.

[66] E. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt Rinehart and Winston, New York, 1976.

[67] B. Lee and D. Brooks. Accurate and Efficient Regression Modeling for Microarchitectural Performance and Power Prediction. In *Architectural Support for Programming Languages and Operating Systems*, pages 185–194, 2006.

[68] B. Lee, K. Chung, B. Koo, and N. Eum. Thermal Sensor Allocation and Placement for Reconfigurable Systems. *Transactions on Design Automation of Electronic Systems*, 4(41):50:1–23, 2009.

[69] J. Lee, K. Skadron, and S. Chung. Predictive Temperature-Aware DVFS. *IEEE Transactions on Computers*, 59(1):127–133, 2010.

[70] K. Lee and K. Skadron. Using performance Counters for Runtime Temperature Sensing in High-Performance Processors. In *Proceedings of the Workshop on High-Performance, Power-Aware Computing*, page 232.1, 2005.

[71] K. Lee, K. Skadron, and W. Huang. Analytical Model for Sensor Placement on Microprocessors. In *Proceedings of the International Conference on Computer Design*, pages 24–30, 2005.

[72] C. Lefurgy and X. Wang. Power Capping: A Prelude to Power Shifting. *Cluster Computing*, 2008.

[73] W. Liao, L. He, and K. Lepak. Temperature and Supply Voltage Aware Performance and Power Modeling at Microarchitecture Level. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(7):1042–1053, 2005.

[74] F. Liu. A General Framework for Spatial Correlation Modeling in VLSI Design. In *Proceedings of the Design Automation Conference*, pages 817–822, 2007.

[75] J. Long, S. Memik, G. Memik, and R. Mukherjee. Thermal Monitoring Mechanisms for Chip Multiprocessors. In *ACM Transactions on Architecture and Code Optimization*, volume 5(2), pages 9:1–9:23, 2008.

[76] G. Magklis, M. L. Scott, G. Semeraro, D. H. Albonesi, and S. Dropsho. Profile-Based Dynamic Voltage and Frequency Scaling for a Multiple Clock Domain Microprocessor. In *Proceedings of the International Symposium on Computer Architecture*, pages 14–25, 2003.

[77] D. Meisner, B. T. Gold, and T. F. Wenisch. The PowerNap Server Architecture. *ACM Transactions on Computer Systems*, 29(1):205–216, 2011.

[78] S. Memik, R. Mukherjee, M. Ni, and J. Long. Optimizing Thermal Sensor Allocation for Microprocessors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(3):516–527, 2008.

[79] K. Meng, R Joseph, and R. P. Dick. Multi-Optimization Power Management for Chip Multiprocessors. In *Proceedings of Parallel Architectures and Compilation Techniques*, pages 177–186, 2008.

[80] F. J. Mesa-Martinez, E. Ardestani, and J. Renau. Characterizing Processor Thermal Behavior. In *Architectural Support for Programming Languages and Operating Systems*, pages 193–204, 2010.

[81] F. J. Mesa-Martinez, M. Brown, J. Nayfach-Battilana, and J. Renau. Measuring Performance, Power, and Temperature from Real Processors. In *Proceedings of the International Symposium on Computer Architecture*, pages 1–10, 2007.

[82] R. Mukherjee and S. Memik. Physical Aware Frequency Selection for Dynamic Thermal Management in Multi-Core Systems. In *Proceedings of International Conference on Computer Aided Design*, pages 547–552, 2006.

[83] R. Mukherjee and S. Memik. Systematic Temperature Sensor Allocation and Placement for Microprocessors. In *Proceedings of the Design Automation Conference*, pages 542 – 547, 2006.

[84] R. Mukherjee, S. Mondal, and S. O. Memik. Thermal Sensor Allocation and Placement for Reconfigurable Systems. In *Proceedings of International Conference on Computer Aided Design*, pages 437–442, 2006.

[85] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, and G. De Micheli. Temperature-Aware Processor Frequency Assignment for MPSoCs Using Convex Optimization. In *CODES+ISSS*, pages 111–116, 2007.

[86] A. N. Nowroz, R. Cochran, and S. Reda. Thermal Monitoring of Real Processors: Techniques for Sensor Allocation and Full Characterization. In *Proceedings of the Design Automation Conference*, pages 56–61, 2010.

[87] J. Parker, R. Kenyon, and D. Troxel. Comparison of Interpolating Methods for Image Resampling. *IEEE Transactions on Medical Imaging*, 2(1):31–39, 1983.

[88] I. Paschalidis, B. Li, and M. Caramanis. A Market-Based Mechanism for Providing Demand-Side Regulation Service Reserves. In *Proceedings of the Decision and Control and European Control Conference*, pages 21 –26, December 2011.

[89] M. Pedram and S. Nazarian. Thermal Modeling, Analysis, and Management in VLSI Circuits: Principles and Methods. *Proceedings of the IEEE*, 94(8):1487–1501, 2006.

[90] M. D. Powell, M. Gomaa, and T. N. Vijaykumar. Heat-and-Run: Leveraging SMT and CMP to Manage Power Density Through the Operating System. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 260–270, 2004.

[91] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1996.

[92] K. Rangan, Gu-Yeon Wei, and D. Brooks. Thread Motion: Fine-Grained Power Management for Multi-core Systems. In *Proceedings of International Symposium on Computer Architecture*, volume 37, 2009.

[93] S. Reda, R. Cochran, and A. K. Coskun. Adaptive Power Capping for Servers with Multi-threaded Workloads. *to appear in IEEE Micro, 2012.*

[94] S. Reda, R. Cochran, and A. N. Nowroz. Improved Thermal Tracking for Processors Using Hard and Soft Sensor Allocation Techniques. *IEEE Transactions on Computers*, 60(6):841–861, 2011.

[95] R. Ribando and K. Skadron. Many-Core Design from a Thermal Perspective. pages 746–749, 2008.

[96] M. J. Roberts. *Signals and Systems*. McGraw Hill, first edition, 2004.

[97] A. Rogalski. Infrared Devices and Techniques. *Optoelectronics Review*, pages 111–136, 2002.

[98] E. Rotem, J. Hermerding, C. Aviad, and C. Harel. Temperature Measurement in the Intel Core Duo Processor. In *Proceedings of the International Workshop on Thermal Investigations of ICs*, pages 23–27, 2006.

[99] M. Ruggiero, D. Bertozzi, L. Benini, M. Milano, and A. Andrei. Reducing the Abstraction and Optimality Gaps in the Allocation and Scheduling for Variable Voltage/Frequency MPSoC Platforms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(3):378 – 391, 2009.

[100] J. Sartori and R. Kumar. Distributed Peak Power Management for Many-Core Architectures. In *Proceedings of Design, Automation and Test in Europe Conference*, pages 1556–1559, 2009.

[101] K. D. Sauer and J. P. Allebach. Iterative Reconstruction of Band-Limited Images from Nonuniformly Spaced Samples. *IEEE Trans. Circuits and Systems*, 34(12):1497–1506, 1987.

[102] S. Sharifi, R. Z. Ayoub, and T. S. Rosing. TempoMP: Integrated Prediction and Management of Temperature in Heterogeneous MPSoCs. In *Proceedings of Design, Automation, and Test in Europe*, pages 593–598, 2012.

[103] T. Sherwood, E.. Perelman, G. Hamerly, S. Sair, and B. Calder. Discovering and Exploiting Program Phases. *Transactions of the International Symposium on Microarchitecture*, 23(6):84–93, 2003.

[104] K. Skadron. Hybrid Architectural Dynamic Thermal Management. In *Proceedings of Design, Automation and Test in Europe*, pages 10–15, 2004.

[105] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *Proceedings of the International Symposium on Computer Architecture*, pages 259–270, June 2003.

[106] C. D. Spradling. SPEC CPU2006 benchmark tools. *ACM SIGARCH Computer Architecture News*, pages 130–134, 2007.

[107] G. Strang. *Computational Science and Engineering*. Wellesly-Cambridge Press, first edition, 2007.

[108] N. Tolia. Delivering Energy Proportionality with Non Energy-Proprotional Systems-Optimizing the Ensemble. In *First Workshop on Power Aware Computing and Systems*, pages 1–6, November 2008.

[109] A. Vincenzi, A. Sridhar, M. Ruggiero, and D. Atienza. Fast Thermal Simulation of 2D/3D Integrated Circuits Exploiting Neural Networks and GPUs. In *International Symposium on Low Power Electronics and Design*, August 2011.

[110] X. Wang and M. Chen. Cluster-Level Feedback Power Control for Performance Optimization. In *Proceedings of the International Symposium on High Performance Computer Architecture*, pages 101–110, 2008.

[111] X. Wang, M. Chen, C. Lefurgy, and T. W. Keller. SHIP: A Scalable Hierarchical Power Control Architecture for Large-Scale Data Centers. *IEEE Transactions on Parallel and Distributed Systems*, 23(1):168–176, 2012.

[112] Y. Wang, K. Ma, and X. Wang. Temperature-Constrained Power Control for Chip Multiprocessors with Online Model Estimation. In *Proceedings of the International Symposium on Computer Architecture*, 2009.

[113] S. J. E. Wilton and N. P. Jouppi. CACTI: An Enhanced Cache Access and Cycle Time Model. *IEEE Journal Solid-State Circuits*, 31(5):677–688, 1996.

[114] J. Winter and D. Albonesi. Addressing Thermal Nonuniformity in SMT Workloads. *ACM Transactions on Architecture and Code Optimization*, 5(1):4:1–4:28, 2008.

[115] W. Wu, L. Jin, J. Yang, P. Liu, and Sheldon X. D. Tan. Efficient Power Modeling and Software Thermal Sensing for Runtime Temperature Monitoring. In *Transactions on Design Automation of Electronic Systems*, volume 12, 2007.

[116] J. Yang, X. Zhou, M. Chrobak, Y. Zhang, and L. Jin. Dynamic Thermal Management through Task Scheduling. In *International Symposium on Performance Analysis of Systems and Software*, pages 191 – 201, 2008.

[117] I. Yeo and Eun J. Kim. Hybrid Dynamic Thermal Management Based on Statistical Characteristics of Multimedia Applications. *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 321–326, 2008.

[118] I. Yeo, C. Liu, and E. Kim. Predictive Dynamic Thermal Management for Multicore Systems. In *Proceedings of the Design Automation Conference*, pages 734–739, 2008.

[119] M. Yuffe et al. A Fully Integrated Multi-CPU, Processor Graphics, and Memory Controller 32-nm Processor. *IEEE Journal of Solid-State Circuits*, 47(1):194–205, 2012.

[120] F. Zanini, D. Atienza, and G. De Micheli. A Control Theory Approach for Thermal Balancing of MPSoC. In *Proceedings of the Asia and South Pacific Design Automation Conference*, pages 37–42, 2009.

[121] S. Zhang and K. S. Chatha. Approximation Algorithm for the Temperature-Aware Scheduling Problem. In *Proceedings of International Conference on Computer Aided Design*, pages 281–288, 2007.

[122] Y. Zhang, A. Srivastava, and M. Zahran. On-Chip Sensor-Driven Efficient Thermal Profile Estimation Algorithms. *ACM Transactions on Design Automation of Electronic Systems*, 15(3):25:1, 2010.

[123] C. Zhu, Z. Gu, L. Shang, R. Dick, and R. Joseph. Three-Dimensional Chip-Multiprocessor Run-Time Thermal Management. *IEEE Trans on Computer Aided Design of Integrated Circuits and Systems*, 27(8):1479–1492, 2008.