Consistent Runtime Thermal Prediction and Control Through Workload Phase Detection

Ryan Cochran
Division of Engineering
Brown University
Providence, RI 02912
Email: ryan_cochran@brown.edu

Sherief Reda
Division of Engineering
Brown University
Providence, RI 02912
Email: sherief_reda@brown.edu

ABSTRACT

Elevated temperatures impact the performance, power consumption, and reliability of processors, which rely on integrated thermal sensors to measure runtime thermal behavior. These thermal measurements are typically inputs to a dynamic thermal management system that controls the operating parameters of the processor and cooling system. The ability to predict future thermal behavior allows a thermal management system to optimize a processor's operation so as to prevent the on-set of high temperatures. In this paper we propose a new thermal prediction method that leads to consistent results between the thermal models used in prediction and observed thermal sensor measurements, and is capable of accurately predicting temperature behavior with heterogenous workload assignment on a multicore platform. We devise an off-line analysis algorithm that learns a set of thermal models as a function of operating frequency and globally defined workload phases. We incorporate these thermal models into a dynamic voltage and frequency scaling (DVFS) technique that limits the maximum temperature during runtime. We demonstrate the effectiveness of our proposed system in predicting the thermal behavior of a real quad-core processor in response to different workloads. In comparison to a reactive thermal management technique, our predictive method dramatically reduces the number of thermal violations, the magnitude of thermal cycles, and workload runtimes.

ACM Categories & Subject Descriptors
B.7.1 [Integrated Circuits]: Types and Design Styles.
General Terms: Design, Performance, Algorithms.
Keywords: thermal prediction, thermal sensing, workload phase, DVFS, proactive control, multicore systems

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 [1]. 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 [15], and thermal cycles create mechanical stresses due to expansions and contractions [1].

Dynamic thermal management (DTM) techniques allow processors to optimize performance while avoiding thermal violations. The most well-known DTM techniques include clock gating, dynamic voltage and frequency scaling (DVFS), and thread migration/scheduling [14, 18, 6, 5]. If none of these techniques succeed in lowering the temperature(e.g., in the case of cooling system failure), most processors are hardwired to "trip" or shut down at a predesignated thermal threshold in order to prevent any subsequent physical damage.

To optimally manage performance in the face of increasingly constrictive thermal thresholds, we propose a novel thermal forecast method that is capable of detecting thermal transitions before their onset. We utilize this forecast in conjunction with a processor's DVFS system to maximize throughput while still avoiding thermal violations during runtime. Our contributions are as follows:

1. Using the measurements of the processor's performance counters, we design a methodology capable of detecting workload phase transitions and associating them with changes in thermal behavior. Due to the relatively slow thermal transient response, and to the immediate correlations between changes in power consumption and performance counter measurements, our method is capable of accurately predicting thermal transitions well before their onset.
2. Our phase analysis method uses principal component analysis together with k-means clustering to learn global workload phases and associate them with thermal models using a large set of representative workload data. Because these phases are globally defined, (i.e., their implications are not restricted to any single application), and because our thermal models capture the thermal coupling between cores, our methodology is capable of forecasting temperature even with heterogenous workload sets operating on multicore systems. To minimize the runtime computational overhead, global phases and thermal models are characterized entirely offline, and the runtime overhead amounts to a table lookup and a few arithmetic operations.
3. We demonstrate the utility of our thermal prediction methodology in a proactive DVFS scheme that effectively prevents ther-

DAC'10, June 13-18, 2010, Anaheim, California, USA

mal violations. Since many processors handle larger thermal violations by abrupt frequency scaling or clock gating, prevention of these thermal violations translates directly to performance gains. In addition, a known thermal trajectory relaxes the need for DVFS throttling as the temperature approaches a thermal violation threshold, thus yielding additional performance gains. Lastly, proactive control increases chip reliability by reducing thermal cycling. We demonstrate that by using thermal forecasting, our proactive throttling scheme eliminates almost all thermal violations while simultaneously reducing the total runtime by 4.7% and the thermal variance by a factor of 2X when compared with an equivalent reactive scheme.

4. While many previous works related to this topic perform validation on pure simulation or hybrid simulation/physical systems, our technique is validated entirely on a real Intel Core i7 940 based workstation running SPEC CPU2006 benchmarks. Our implementation uses real temperature values measured by on-chip embedded sensors, and manages the workstation thermal behavior through DVFS control.

The rest of this paper is organized as follows. We start with an overview of related work and further motivation for our work in Section 2. We provide an overview of our methodology in Section 3, where we describe our off-line phase analysis and model learning methods in Subsection 3.1, and in Subsection 3.2 we describe our runtime phase identification and proposed DVFS method. We provide a comprehensive set of experimental results on a real system in Section 4. Finally, Section 5 summarizes the main conclusions drawn from this work.

2. RELATED WORK AND MOTIVATION

There have been a number of thermal prediction and management techniques proposed recently in the literature [12, 11, 4, 19, 3]. These techniques generally fit a mathematical model to a local window of observed temperatures arising from the execution of an application. The model coefficients are estimated to give the total least square errors between the model results and the actual observed temperatures. To cut down the computational complexity required to update the model coefficients, recursive least square estimation can be used to track changes within and across applications [19]. After being learned, a thermal model can be used to extrapolate future temperatures. Previous approaches utilize different mathematical models for temperature. For example, Coskun *et al.* [4, 3] use an autoregressive moving average model, while Yeo *et al.* [19] use a generic polynomial model. Previous approaches also differ in how the temperature measurements are obtained. In processors that do not support thermal sensors, previous approaches [4, 3, 12] use performance counters and core utilization metrics to estimate the temperature. Such indirect approaches introduce significant inaccuracies as they require extensive power and thermal modeling. Other approaches [19] eliminate the need for such modeling by directly acquiring the temperatures from thermal sensors which are available in most modern processors.

Previous approaches provide excellent temperature tracking as long as the temperature is behaving consistently. When an application's thermal behavior changes during execution, it can deviate significantly from a static model prediction; consequently, the coefficients of the model are no longer accurate and must be updated to reflect the change in the workload behavior. During that time gap when the model coefficients are being adjusted, any model predictions are *inconsistent* with the die temperatures and consequently thermal violations can occur. Previous approaches [3] utilize statistical techniques to detect such drifts in thermal model predictions and re-trigger model learning accordingly. Rather than detecting model drifts after thermal inconsistencies are observed, our work advocates a *consistent* thermal prediction approach that predicts well in advance changes in thermal behavior and accordingly adjusts the thermal model without incurring any drifts or inconsistencies in temperature prediction. In addition, ours is the first work on thermal prediction and proactive control to handle thermal coupling between cores running a heterogenous set of workloads on a real multicore system.

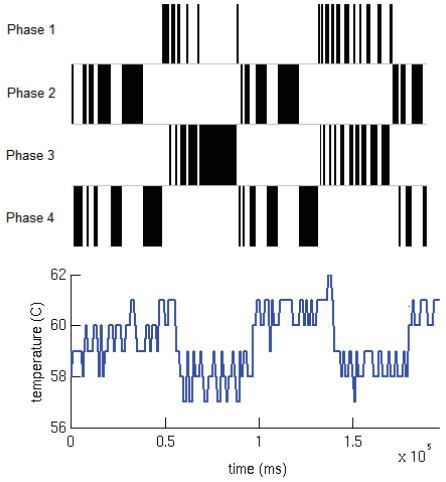

Figure 1: Workload phases and their impact on temperature as measured by the processor's thermal sensors. Different phases trigger different thermal behavior.

3. PROPOSED PHASE-AWARE THERMAL PREDICTION METHODOLOGY

At the highest level, our phase-aware thermal prediction approach takes raw performance counter data that is periodically measured for each core during workload operation and translates this data into a temperature projection for some interval into the future using the concept of workload phases. In this context, a *phase* is a stage of execution in which a workload exhibits near identical power, temperature, and performance characteristics. Because workloads tend to execute in repetitive, predictable patterns [16, 17, 10, 8, 2], workloads progress through these phases in predictable patterns. For example, Figure 1 shows the workload phases of the `tonto` workload during runtime. The top part of the figure shows the workload phase as a function of time, while the bottom part of the figure shows the temperature. This figure clearly shows the program phases executing in a regular pattern with consistent temperature behavior within each phase.

In order to define workload phases and capture temperature dynamics within them in a computationally efficient manner, we propose the methodology that is illustrated by Figure 2. Our method is split between an *off-line analysis* component and a *runtime analysis* component. In the off-line analysis component, performance counter measurements from representative workloads are first fed to a principal component analysis module that reduces the large space of measurements into a smaller space. This smaller space reduces the requirements for the subsequent clustering and modeling stages. The clustering stage computes the centroids of the different workload phases in a global fashion. Using this information, a state-space temperature model is used to learn the relationships between temperature, workload phase and operating fre-

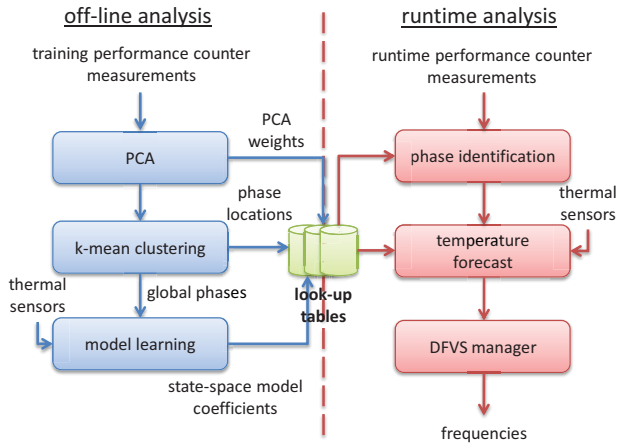

Figure 2: Overall flow of our proposed method.

quency. The complete details of the off-line analysis component are given in Section 3.1. Phase-aware control strategies are performed in a lightweight runtime component. In the runtime component, performance counter measurements are first mapped into principal component space using the component weights calculated in the off-line analysis part. Afterwards the current workload phase is identified and the corresponding state-space model is pulled from the lookup-table. The model is used to forecast the temperature dynamic associated with the current phase. The runtime component select the highest operating frequency that does not lead to thermal violations. The complete details of the run-time analysis component are given in Section 3.2.

3.1 Offline Thermal Phase Analysis

In order to avoid excessive runtime overhead, global phase analysis and within-phase temperature modeling are performed offline using data generated for a set of representative workloads. This data includes raw performance counter numbers sampled from each core at regular intervals, as well as embedded thermal diode sensor measurements for each core. Prior to phase analysis, the raw performance counter data is normalized by the number of clock cycles that have executed over the same interval. In our analysis we consider the measurements obtained from the performance counters at an instance of time i as an *observation* that is mathematically represented as a random vector $\mathbf{x_i}$. If there are p performance counters then the observation measured at instance i can be represented as vector $\mathbf{x_i} = (x_{i1}, \ldots, x_{ip})$.

In many-core processors, performance counters could reach a total of tens and hundreds in numbers which can slow down the off-line analysis and the subsequent runtime thermal management.

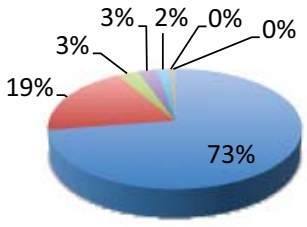

Figure 3: Percentage of information (as measured by the statistical variance) captured by principal components.

In reality many of these counters exhibit correlations in their measurements (e.g., the cache miss performance counter is correlated to the retired instructions performance counter). Thus, we propose to use *principal component analysis* (PCA) as a method to reduce the computational effort. PCA transforms a number of correlated variables into a smaller number of uncorrelated variables, or *principal components*, while retaining most of the original information [9]. The smaller number of uncorrelated variables can then be used for all workload phase analysis. For example, we compute the principal components of 47 performance counter variables of the Core i7 processor; Figure 3 gives the contribution (as measured by the statistical variance) of the principal components towards explaining the observed measurements. The chart shows that a handful of principal components can capture the information of all performance counters. PCA essentially transforms an observation $\mathbf{x_i} = (x_{i1}, \ldots, x_{ip})$ to $\mathbf{x'_i} = (x'_{i1}, \ldots, x'_{id})$, where $d << p$ with negligible loss off information.

Within the d-dimensional space defined by the principal component axes, we use k-means clustering to define the *global phase* locations that most closely approximate the n observations in the representative workload data. k-means clustering finds a partition of the n observations into $k < n$ sets, $S_1, S_2, \ldots, S_k$, to minimize the sum of squares,

$$\min \sum_{j=1}^{k} \sum_{\mathbf{x_i} \in S_j} ||\mathbf{x'_i} - \mu_\mathbf{j}||^2, \quad (1)$$

where $\mu_\mathbf{j}$ is the mean of the observations in set S_j. Optimal k-means clustering is an NP-hard problem; therefore, all algorithm implementations are heuristic and are not guaranteed to converge to a global optimum. We use the standard k-means algorithm which 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 [13]. To find the best number of clusters (or equivalently phases), we increase gradually the number of phases/clusters k and repeat k-means algorithm until the error percentage defined by the sum of squares normalized to the observation magnitudes is less than 5%. For example, Figure 4 gives the observations obtained from the CPU SPEC06 workloads (we display only the values of the first two principal components) and the centers of clusters of these observations as computed using the k-means algorithm.

The overall effect of applying PCA and k-means clustering to the performance counter data is that the description of a performance counter sample set with thousands of samples in potentially hundreds of dimensions is reduced to a small number of points in a low-dimensional space. This dramatic reduction greatly simplifies temperature modeling and permits negligible runtime overhead with very accurate and consistent temperature prediction results.

There is an important distinction between our use of performance counter measurements and previous works [11, 3]. Previous works did not make use of thermal sensors; in contrast, we utilize the temperature data collected from each core's thermal sensor to *learn* a state-space temperature model associated with each global phase. Our choice of the state-space model is a natural one because it captures accurately the physical dynamics of the temperature as a function of power consumption. However, our state-space model *does not* model power consumption; instead, it simply learns the relationship between temperature, performance counters, and operating frequencies from the provided training data. Because performance counters measure processor switching events, and because processor power dissipation is linearly related to the degree of switching activity within a given operating voltage and frequency, we assume

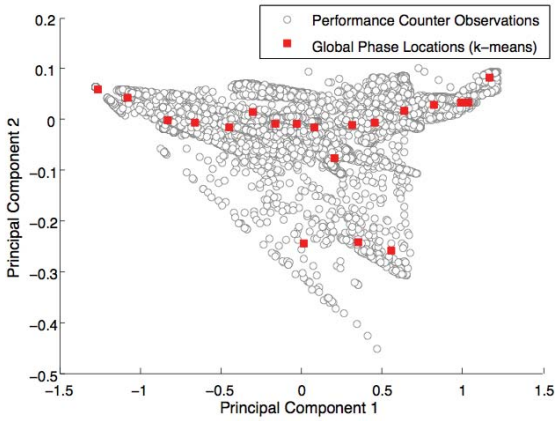

Figure 4: First two principal components together with the centroids of the clusters of the performance counter measurements of SPEC06 workloads.

that the processor power dissipation will remain constant within a performance counter phase at a particular frequency. Thus, the continuous time invariant state-space equation can be approximated over a small time interval with a simplified discrete time-invariant state-space equation. In state-space modeling, the projected temperature for core i at instant $k+1$ is a linear function of its own temperature and the temperature of the other $N-1$ cores at instant k according to

$$T_i[k+1] = \sum_{j=1}^{N} a_{ij} T_j[k] + b_i(f,p), \qquad (2)$$

where $b_i(f,p)$ is a function of the current phase p and frequency f. Because a_{ij} captures the thermal coupling between cores as a function of their physical proximity, it is modeled to be independent of both frequency and phase. The a_{ij} values are learned by observing the *natural response* of idle cores while running thermally aggressive workloads on adjacent cores. For core i in an idle state, $b_i(f,p)$ is assumed to be zero, and the values of a_{ij} are learned by performing least squares regression on the temperatures observed at instant k and $k+1$. Once the values of a_{ij} have been obtained, the values of $b_i(f,p)$ for each frequency and phase are learned using least squares regression on the phase designations calculated for representative workloads gathered at each operating frequency. By modeling the thermal coupling between cores and defining workload phases globally, this methodology is well equipped to handle heterogenous workload sets running on multicore systems.

3.2 Runtime Thermal Prediction and Control

Given the phase-dependent state-space model, our runtime predictive thermal manager projects the future processor temperatures associated with the current phase under different frequency assignment scenarios. The thermal management unit then chooses the highest frequency that does not lead to a future thermal violation.

With the PCA coefficients and the phase cluster locations calculated for representative workloads during offline analysis, runtime thermal prediction and control are relatively lightweight tasks. This is ideal because the phase-aware thermal control strategy must not incur computational overhead that offsets any performance improvements offered by predictive thermal management. The runtime thermal prediction component activates periodically during workload operation, taking raw performance counter and temperature data and outputting the projected temperatures for each core at each processor frequency.

In between activations of the thermal prediction component, the performance counter values for each core are normalized to CPU cycles and buffered. Just prior to thermal prediction, these measurements are averaged and rotated into the PCA space. This task is greatly simplified by the fact that the weights used in this rotation are already calculated during off-line analysis, and thus the result is a weighted linear combination of the performance counter measurements. Once in PCA space, the data observations are classified according to clustering results. Since the locations of the k phase clusters have already been calculated in the offline component, this step amounts to finding the distance between a data observation and the k phase locations in PCA space, and classifying the observation according to the closest phase location.

Once a phase has been determined for each core, the state-space model $a(\cdot,\cdot)$ and $b(\cdot,\cdot)$ coefficients of the phase are retrieved and the temperature of each core is projected according to Equation (2). The maximum projected temperature of all the cores at each possible operating frequency is used as input to the temperature control strategy.

A typical reactive thermal control scheme will incorporate temperature thresholds that are designed to prevent thermal emergencies and subsequent damage to the processor. As the processor temperature crosses these thresholds, the CPU performance is throttled back in order to bring the temperature under control. Higher temperature thresholds generally induce higher performance penalties, as crossing them indicates a more urgent thermal situation. If a workload's temperature profile regularly crosses these higher temperature thresholds, the workload will experience severe performance penalties which can significantly increase its overall runtime. For example, the Core i7 processor includes Thermal Control Circuit (TCC) unit in hardware that reactively throttles the operating frequencies and voltages based on two activation temperature thresholds (`T_activation`, `T_activation` + 5 °C) [7]. When the maximum observed processor temperature exceeds `T_activation`, the TCC will reduce the processor frequency by one clock frequency ratio. If after 1 ms the temperature does not recover to below `T_activation`, the TCC continues to reduce the frequencies and voltages. If the temperature exceeds `T_activation` + 5 °C, however, clock gating is initiated regardless of whether the lowest frequency and voltage have been reached.

However, if the thermal control system has an idea of the maximum frequency that can be set without crossing the higher thermal thresholds, these severe performance penalties (e.g., clock gating) can be avoided altogether. With the inherent delay between the performance counter values and the resulting temperature fluctuations, as well as the thermal predictions offered by performance counter phase analysis, our thermal control strategy anticipates these thermal crossings well before they occur. Our thermal control strategy sets the operating frequency to be the highest value for which the maximum projected temperature does not exceed the high penalty thermal threshold.

4. EXPERIMENTAL RESULTS

A. Experimental Infrastructure. Our experimental infrastructure consists of two main components: a phase-aware thermal characterization component and a runtime phase-aware DVFS policy. In the offline component, optimal global phases are defined and within-phase thermal models are learned through extensive analysis of a large body of temperature and performance counter characterization data collected from a representative set of workloads at various

frequencies/voltages settings. The results of this offline analysis are stored in tabular matrix form for lookup by the runtime component. The runtime component uses these offline-generated matrices to calculate the workload phases in real-time using incoming performance counter data, and performs intelligent DVFS using accurate within-phase thermal predictions. Our experimental setup is as follows.

- All offline characterization and runtime experiments are performed on an Intel Core i7 940 45 nm processor, running a 2.6.10.8 Linux kernel OS.
- Performance counter data is collected using the `pfmon` (version 3.9) command-line utility, which is configured to sample system-wide performance counter measurements with per-core granularity at 100 ms intervals. The performance events that are counted include those deemed most likely to contribute to the maximum core temperature (instructions retired, floating point instructions executed, and the number of conditional instructions executed).
- Each core within the Core i7 processor is equipped with a digital thermal sensor, which registers the maximum temperature observed by a set of thermal sensor diodes which are embedded within each core. Temperatures are reported with per-core granularity at 100 ms intervals by interfacing the `pfmon` utility with the functionality of the Linux `lm-sensors` hardware-monitoring package.
- The Core i7 processor operating frequencies are accessed using the Linux `cpufreq` library functionality, and are set to 1333 MHz intervals within the range 1.60 GHz − 2.93 GHz. The processor operating voltages are automatically set in hardware in correspondence with the processor's operating frequency. The operating period for both reactive and phase-aware strategies is 100 ms and synchronizes with the incoming temperature data.
- All workloads are drawn from the SPEC CPU2006 benchmark suite. A large number of both integer and floating point benchmarks are used as representative workloads during offline analysis. We assess our phase-aware methodology for each of the following: `astar`, `bzip2`, `gcc`, `dealII`, `calculix`, `bwaves`, and `tonto`. These workloads are selected because they exhibit widely varying temperature profiles that are interesting for thermal prediction. Within each of these experiments, separate instances of the selected benchmark are run simultaneously on each core. In addition, two experiments are performed with a mixed set of workloads. In `mix1`, each core is assigned one of `bzip2`, `bwaves`, `dealII`, or `povray`, while `sjeng`, `gcc`, `perlbench`, or `tonto` are selected for `mix2`.

B. Experiments. Our analysis of the global performance counter measurements reveals that there are 15 global phases as computed by k-means clustering on the first 2 principal components. We use these 15 global phases and the corresponding temperature measurements at different frequencies in order to train state-space models for each phase as given by Equation (2). In our first experiment we assess the accuracy of our global state-space models in projecting the temperatures of the different SPEC06 workloads. Figure 5 gives the average error percentage for selected workloads at various frequency settings. The results show that our state-space modeling approach accurately predicts the temperature with around 1% error for homogenous workload sets. While the estimation error is higher for the heterogenous case, it still shows high accuracy with between 2 − 2.5% error. Figure 6 illustrates successful temperature tracking for `mix1`. It is observed that the estimation accuracy

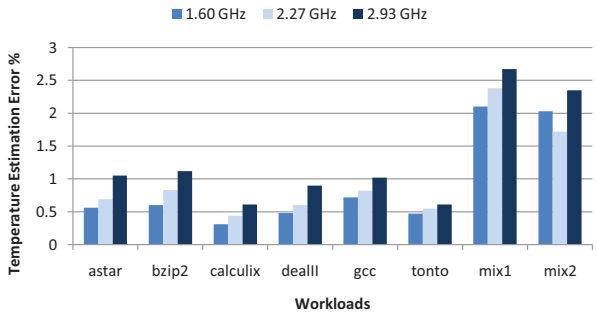

Figure 5: Thermal estimation error (in %) as computed by our learned state-space models.

of the state-space model tends to be less accurate as the operating frequency increases. The reason for this behavior is that higher frequencies produce higher temperatures and corresponding leakage power which introduces non-linearities into the system, thus making the linear state-space model less accurate. These results confirms the accuracy of our methodology in tracking temperature with mixed thermally aggressive workloads running on a multicore system.

In our second experiment we compare our runtime predictive method with a reactive thermal management method that adjusts the processor's frequency only in response to a thermal violation. We designed the reactive strategy to be similar to Intel's thermal management policy, but with lower thresholds so as to provoke thermal violations. We set two temperature threshold limits: the *soft* temperature limit is set to 51°C and a second *hard* temperature limit is set to 55 °C. For each operating period during which the maximum temperature exceeds the soft limit but remains below the hard limit, we incrementally lower the processor frequency by one clock ratio. If the maximum temperature exceeds the hard limit, however, the processor frequency is set to the lowest possible level for 5 seconds. In Table 1 we compare our phase-aware thermal prediction technique to the reactive strategy. We report the number of thermal violations above the hard limit, the temperature variance which is an indicator of thermal oscillations, and finally the improvement in total runtime. The results show that our technique drastically cuts down the number of hard limit violations. The cause of the runtime improvement is twofold. The reduction in the number of violations preempts the thermal management unit from executing drastic power-reduction mechanisms

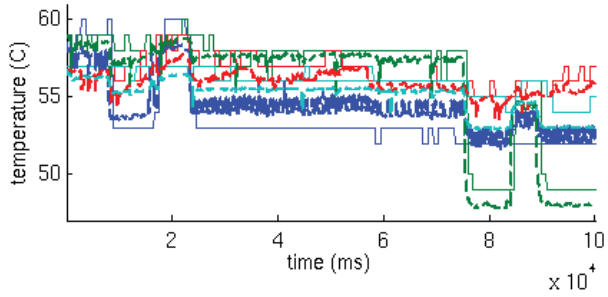

Figure 6: Temperature tracking for section of `mix1` execution. The dotted lines represent temperature estimates for each core while the solid lines represent the measured temperatures.

	astar	bzip2	gcc	dealII	tonto	calculix	mix1	mix2	mean
Reactive Thermal Violations	14	2	8	8	1	24	70	15	17.8
Phase-Aware Thermal Violations	2	0	0	1	0	0	0	0	0.38
Reactive Temperature Variance	0.96	2.2	1.94	1.00	0.45	0.75	3.07	3.11	1.65
Phase-Aware Temperature Variance	0.64	1.5	1.17	0.63	0.17	0.25	1.38	1.03	0.84
Phase-aware Runtime Improvement	6.45%	5.85%	0.96%	4.12%	4.50%	1.00%	6.90%	7.60%	4.67%

Table 1: Comparison of predictive and reactive thermal management methods using a representative SPEC06 workloads.

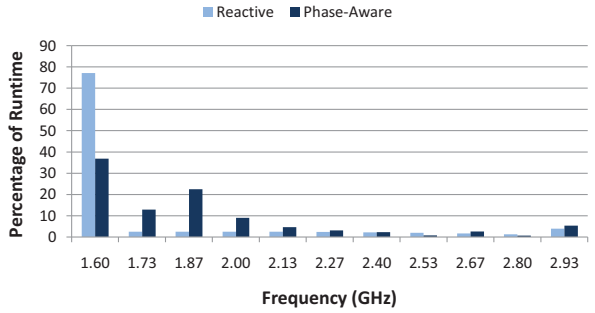

Figure 7: Percentage of time spent at each frequency as a function of reactive method and proposed method.

which improves overall throughput. In addition, a known thermal trajectory relaxes the need for cautious DVFS throttling as the temperature approaches the hard limit, thus yielding additional performance gains.

We also track the percentage of time the core spent at each frequency for both the reactive method and our phase-aware method and plot the results in Figure 7. The results show that the thermal violations force the reactive method to use lower frequencies for a larger percentage of time, where our method is capable of utilizing higher frequencies for more periods of time. This improved frequency distribution leads to better overall throughput.

5. CONCLUSIONS

In this paper we propose a novel thermal prediction technique and incorporate it into a proactive CPU throttling scheme designed to reduce thermal violations. Our thermal prediction scheme, which relies on global workload phase analysis and state-space thermal models, incurs minimal runtime overhead while accurately predicting temperature even for heterogenous workload sets running on a multicore system. Our thermal prediction methodology is incorporated into proactive DVFS throttling scheme that can estimate the impact of different frequency choices, and thereby select the highest frequency that does not lead to thermal violations. We implement our thermal prediction and DVFS techniques on a real system with a state-of-the-art processor. Our results demonstrate the superiority of our approach in preventing thermal violations, lowering thermal cycles, and improving total runtime relative to an equivalent reactive scheme.

Acknowledgments: This work is partially supported by a NSF CAREER grant number 0952866.

6. REFERENCES

[1] 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.

[2] C.-B. Cho and T. Li. Complexity-based program phase analysis and classification. In *International Conference on Parallel Architectures and Compilation techniques*, pages 105–113, Sep 2006.

[3] 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.

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

[5] 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 *International Symposium on Low Power Electronics and Design*, pages 219–224, 2007.

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

[7] B. Inkley. A New Processor Temperature Specification: Using the Digital Thermal Sensor. In *Intel Developer Forum*, 2008.

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

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

[10] 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.

[11] A. Kumar, L. Shang, L. Peh, and N. Jha. System-Level Dynamic Thermal Management for High-Performance Microprocessors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 27(1):96–108, 2008.

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

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

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

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

[16] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically Characterizing Large Scale Program Behavior. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 45–57, 2002.

[17] T. Sherwood, E. Perelman, G. Hamerly, S. Sair, and B. Calder. Discovering and Exploiting Program Phases. *IEEE Micro*, 23(6):84 – 93, 2003.

[18] W. Wu, L. Jin, Y. Yang, P. Liu, and S. X.-D. Tan. Efficient Power Modeling and Software Thermal Sensing for Runtime Temperature Monitoring. *ACM Transactions on Design Automation of Electronic Systems*, 12(3):1–29, 2007.

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