Thermal Prediction and Adaptive Control Through Workload Phase Detection

RYAN COCHRAN and SHERIEF REDA, Brown University

Elevated die temperature is a true limiter to the scalability of modern processors. With continued technology scaling in order to meet ever-increasing performance demands, it is no longer cost effective to design cooling systems that handle the worst-case thermal behaviors. Instead, cooling systems are designed to handle typical chip operation, while processors must detect and handle rare thermal emergencies. Most processors rely on measurements from integrated thermal sensors and dynamic thermal management (DTM) techniques in order to manage the trade-off between performance and thermal risk. Optimal management requires advanced knowledge of the thermal trajectory based on the current workload behaviors and operating conditions. In this work, we devise novel workload phase classification strategies that automatically discriminate among workload behaviors with respect to the thermal control response. We incorporate workload phase-detection and thermal models into a dynamic voltage and frequency scaling (DVFS) technique that can optimally control temperature during runtime based on thermal predictions. We demonstrate the effectiveness of our proposed techniques in predicting and adaptively controlling the thermal behavior of a real quad-core processor in response to a wide range of workloads. In comparison with state-of-the-art model predictive control (MPC) techniques in previous works on thermal prediction, we demonstrate a 5.8% improvement in instruction throughput with the same number of thermal violations. In comparison with simple proportional-integral (PI) feedback control techniques, we improve instruction throughput by 3.9%, while significantly reducing the number of thermal violations.

Categories and Subject Descriptors: B.7.1 [Integrated Circuits]: Types and Designs
General Terms: Design, Performance, Algorithms

ACM Reference Format:
DOI = 10.1145/2390191.2390198 http://doi.acm.org/10.1145/2390191.2390198

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 hotspots that severely limit overall system performance, as the speed of transistors and interconnects are negatively affected by temperature [Brooks et al. 2007]. 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 [Pedram and Nazarian 2006], and thermal cycles create mechanical stresses due to expansions and contractions [Brooks et al. 2007].

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 [Hamann et al. 2007]. 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. 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. Traditional air-cooling and even advanced liquid-cooling systems are unable to efficiently handle the worst-case power dissipation and must be augmented with additional thermal mitigation strategies [Sabry et al. 2011; Coskun et al. 2010]. 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 [Mukherjee and Memik 2006; Wu et al. 2007; Herbert and Marculescu 2007; Hamann et al. 2007; Coskun et al. 2008], 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 trade-off 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 setpoint, 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 setpoint incur only incremental performance penalties. Each threshold above the setpoint, or hard threshold, 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 setpoint (soft limit) and the emergency level constitutes the thermal guard band. The upper portion of Figure 1 illustrates a sample scheme employed by many popular commercial processors [Berktold and Tian 2010], 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 of 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 [Han et al. 2006; Brooks et al. 2007; Wang et al. 2009; Sharifi et al. 2012; Zanini et al. 2009; Kumar and Atienza 2010; Vincenzi et al. 2011; Coskun et al. 2008, 2009; Kumar et al. 2006, 2008; Lee et al. 2010; Yeo and Kim 2008]. By increasing the
Thermal Prediction and Adaptive Control Through Workload Phase Detection

Fig. 1. Illustration of the reactive DTM control strategy employed by the Corei7 processor compared to a predictive approach.

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 (1) is a standard model used for a variety of runtime control objectives in the literature [Han et al. 2006; Brooks et al. 2007; Wang et al. 2009; Sharifi et al. 2012; Zanini et al. 2009; Kumar and Atienza 2010; Vincenzi et al. 2011].

\[ T_m[i + 1] = \sum_{n=1}^{N} a_{mn} T_n[i] + a_{m}^{idle} + \widehat{g}(s, \ x_m[i]). \] (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 \( x_m \) 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 \( \hat{g}(\cdot, \cdot) \) with appropriate granularity using workload phase detection. We define workload phases as a function of processor performance counters, which measure architectural events (instructions executed, cache 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 Intel Core i7 940-based workstation running heterogeneous 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. In comparison to our previous work on workload phases [Cochran and Reda 2010], we show a 1% increase in performance 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 [Coskun et al. 2009], we demonstrate a 2.9% improvement in performance and a 97% reduction in thermal violations. In comparison to state-of-the-art model predictive control (MPC) techniques [Bartolini et al. 2011, Bartolini et al. 2012], we demonstrate a 5.8% improvement in performance with the same number of thermal violations. Finally, we compare with a simple proportional-integral (PI) technique and demonstrate a 3.9% performance improvement with a 94% reduction in thermal violations.

The rest of this article 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. We go on to describe our phase classification approaches in Section 4, and in Section 5 we describe our predictive DVFS control strategy. We provide a comprehensive set of experimental results on a real quad-core system in Section 6. Finally, Section 7 summarizes the main conclusions drawn from this work.

2. BACKGROUND AND MOTIVATION

The thermal control response is highly workload dependent, 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 [Berktold and Tian 2010]. 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 of 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. 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 [Coskun et al. 2008, Coskun et al. 2009]. 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 [Yeo et al. 2008].

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 the 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 linear combination performance counters and core utilization metrics to estimate the temperature [Kumar et al. 2006, 2008; Lee et al. 2010]. These works are useful as low-cost alternatives to thermal simulation techniques when there are no/limited 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 (1) has emerged in recent literature as a standard thermal model for runtime DTM. It is a natural choice for modeling temperature, because it approximates continuous first-order differential equations that govern transient thermal behavior [Han et al. 2006; Brooks et al. 2007; Wang et al. 2009; Sharifi et al. 2012; Zanini et al. 2009; Kumar and Atienza 2010, Vincenzi et al. 2011; Cochran and Reda 2010]. It explicitly models thermal time constants and thermal coupling among thermal nodes with the coefficients $a_{m}$. It also explicitly models the workload thermal contribution $g(\cdot, \cdot)$ as a function of the control setting and workload characteristics. The approaches in the literature differ
in the form of voltage, frequency, and workload dependence. Several studies use the relationship \( \hat{g}(\cdot, \cdot) \propto f^\alpha \), where \( \alpha \) is a parameter between 1 and 2 depending on how the voltage scales with frequency [Yeo and Kim 2008]. Other studies relate \( \hat{g}(\cdot, \cdot) \) to frequency \( f \), supply voltage \( V_{dd} \), and workload cycles-per instruction (CPI) using the following empirical formula.

\[
\hat{g}(\cdot, \cdot) = k_0 V_{dd}^2 f + k_0 + (k_c + k_d f) CPI[i]^b,
\]

where \( k_0 - k_c \) are parameters that must be learned [Bartolini et al. 2011, 2012].

Like these works, we choose the discrete time-invariant state-space model for thermal control, because it explicitly models thermal time constants, thermal coupling, and power consumption as a function of control setting and workload. 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 somewhere between linearly and quadratically with frequency for a subset of the DVFS settings, and remain constant for other frequencies. Assuming a fixed exponent for \( f \) can introduce significant error, in this case. To counter this, we calculate a distinct value for \( \hat{g}(\cdot, \cdot) \) for each DVFS setting \( s \).

In a similar vein, we contend that assumptions about the relationship between \( \hat{g}(\cdot, \cdot) \) and the 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 hotspot 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. Our approach leverages the fact that workloads within a local window of performance counter values have similar power values by defining **workload phases**. We develop classifiers that partition the space of workload behaviors into phases, and we calculate a distinct value of \( \hat{g}(\cdot, \cdot) \) for each DVFS setting within each partition. 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.

Workload phase classification is a well-established technique for representing the average behavior of a workload over a local time window [Sherwood et al. 2003; Isci et al. 2006; Cho and Li 2006]. In our previous work [Cochran and Reda 2010], we use the \( K \)-means clustering algorithm employed in state-of-the-art phase classification tools [Sherwood et al. 2003] in order to partition the performance counter input space. We reduce the dimensionality of our performance counter inputs using principal component analysis (PCA) in order to reduce the number of classifier parameters. In comparison to our previous work, this work develops a supervised model learning technique using expectation-maximization (EM) and multinomial logistic regression (MLR) in order to iteratively redefine thermal phase boundaries based on the thermal model estimates.

3. METHODOLOGY

At the highest level, our thermal prediction and control strategy divides between offline analysis and runtime control. 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 outputs the probability of being in each phase for each core, and these probabilities are subsequently used to calculate the expected value of the per-core thermal model predictions. 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 4. 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.

As stated in Section 2, we learn \( \hat{g}(\cdot, \cdot) \) in Equation (1) for each combination of 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 thermally-coupled adjacent cores. We expose heating and cooling of idle cores by throttling the DVFS setting every five seconds such that every possible transition is exercised. By assuming that the value of \( \hat{g}(\cdot, \cdot) \) 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 \( \hat{g}(\cdot, \cdot) \) for each phase and DVFS setting to approximate the training data observations \( g_m[i] \) defined in Equation (3).

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

A detailed description of our phase classification techniques is given in Section 4. 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 [Johnson and Wichern 2007]. Before applying PCA, we subtract the mean values and normalize the inputs in order to account for the different magnitudes among performance counters. The input to the phase classifier is then the first two principal components of the performance counter values.

4. WORKLOAD PHASE CLASSIFICATION

It is well established that workloads execute in consistent and repetitive patterns [Sherwood et al. 2003; Isci et al. 2006; Cho and Li 2006], 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 [Sherwood et al. 2003; Isci et al. 2006]. Other works use metrics derived from performance counters, such as IPC and cache misses per cycle [Isci et al. 2006; Cochran and Reda 2010]. 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 third, 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 we explored in our previous work [Cochran and Reda 2010]. 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.

4.1. Phase Classification Using K-Means Clustering

A standard technique for defining workload phases in the literature is $K$-means clustering [Sherwood et al. 2003]. In our previous work [Cochran and Reda 2010], 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 per-core 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. After assigning a phase to each core, the corresponding thermal model for each core is 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 [Lawler 1976]. An average value is calculated for $g_m[i]$ for each phase and the DVFS setting combination. At runtime, we select the average value corresponding to the current phase and future setting to be the estimate for $\hat{g}(\cdot, \cdot)$ for each core.

4.2. 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 $K$, 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 (4), which maps a vector of $K$ continuous values to a set of $K$ discrete probabilities that sum to 1. Each vector value is then parameterized with the dot product of a $d$-dimensional set of logistic model weights $w_k$, which are learned as a function of the desired phase probabilities, and the phase classifier inputs $x_m[i]$. In order to simplify notation, we assume that $x_m[i]$ includes a constant term.

$$\pi_{mki} = \frac{\exp(w_k \cdot x_m[i])}{\sum_{k'=1}^{K} \exp(w_{k'} \cdot x_m[i])}. \quad (4)$$

The variable $\pi_{mki}$ denotes the probability that sample $i$ for core $m$ is in phase $k$ given input $x_m[i]$. In Equation (5), we define $\phi_{nki}$ to be the likelihood of observing output $g_m[i]$ with setting $s$ in phase $k$, given the expected value $\mu_{sk}$ and error variance $\sigma_{sk}^2$, both of which are model parameters that must be estimated.

$$\phi_{nki} = N \left( g_m[i] \mid \mu_{sk}, \sigma_{sk}^2 \right). \quad (5)$$

We also introduce a binary variable $z_{nki} \in \{0, 1\}$ 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,

$$L(Z) = \log \prod_{m=1}^{N} \prod_{k=1}^{K} \prod_{i=1}^{M} (\pi_{mki} \phi_{nki})^{z_{nki}}, \quad (6)$$

where $Z$ 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 $E[Z]$ using the $K$-means output. The logistic weights $w_k$ and values for $\mu_{sk}$ and $\sigma^2_{sk}$ are then estimated such that $L(E[Z])$ is maximized. The value of $E[Z]$ 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 2, until $L(Z)$ converges within a predefined tolerance. Each EM iteration is summarized with the following.

**Maximization Step.**

$$\left[ w_k, \mu_{sk}, \sigma^2_{sk} \right]_{v_k,k} = \text{argmax} \ L(E[Z]). \quad (7)$$

**Expectation Step.**

$$E[z_{mki}] = \frac{\pi_{mki} \phi_{mki}}{\sum_{k=1}^K \pi_{mki} \phi_{mki}}. \quad (8)$$

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

$$\hat{g}(s, x_m[i]) = \sum_{k=1}^K \pi_{mki} \mu_{sk}. \quad (9)$$

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 is determined to be equiprobable for phase input $x_m[i]$, 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. Our entire proposed offline and online approach using EM is illustrated in Figure 3.

**5. RUNTIME CONTROL**

All of the information learned during offline analysis is passed to the runtime control algorithm in the form of lookup tables, as shown in Figure 3. Periodically during runtime, the DVFS control unit makes a control decision as a 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 3. The input averages, normalization constants, and principal component loadings are conveyed in the lookup tables.
Once transformed, the first two principal components, in addition to a constant term, are used in conjunction with the MLR weights for phase identification according to Equation (4), which estimates the probability of a core being in a particular phase. The value of $\hat{g}(\cdot, \cdot)$ for each core at each DVFS setting is computed as a weighted sum, as in Equation (9).

The DVFS control technique uses thermal predictions in order to maximize performance within a given thermal threshold. The maximum core temperature prediction is queried as a function of each DVFS setting using the state-space model in Equation (1) and the workload thermal contribution $\hat{g}(\cdot, \cdot)$ computed for each core. 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 the following.

$$s_{sel} = \arg\max_s f(s) : \max_m (T_m[i + 1]) < T_{set}. \quad (10)$$

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 sample update period on our test platform.

6. 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 $x_m[i]$ are identified in Figure 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 DVFS settings are manipulated with the cpufreq utility, and the available frequency-voltage settings are $\{1.60\text{GHz}, 0.844V\}$, $\{1.73\text{GHz}, 0.844V\}$, $\{1.87\text{GHz}, 0.844V\}$, $\{2.00\text{GHz}, 0.844V\}$, $\{2.13\text{GHz}, 0.8625V\}$, $\{2.27\text{GHz}, 0.89375V\}$, $\{2.40\text{GHz}, 0.91875V\}$, $\{2.53\text{GHz}, 0.9375V\}$, and $\{2.67\text{GHz}, 0.9875V\}$. 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 lm-sensors library 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: $SK$ phase model parameters $\mu_{sk}$ and $\sigma_{sk}$, where $S$ is the number of DVFS settings and $K$ is the number of phases; $N(N+1)$ state-space model parameters $a_{mn}$ and $a_{m}^{idle}$ combined where $N$ is the number of cores; and $K(C+1)$ classifier weights $w_k$, where $C$ is the number of principal components used.

6.1. 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 one to four instances in order to expose the thermal response to changes in processor utilization. At the same time, we systematically throttle the DVFS setting every five seconds such that every transition is exercised. Each training data sample includes per-core values for the 12 performance counters listed in Figure 6, as well as per-core temperature measurements and the current DVFS setting. By applying PCA, we are able to reduce the set of of 12 performance counter phase classifier inputs to 2 principal components, while retaining the majority of observed variance. We learn the state-space model coefficients $a_{mn}$ and $a_{m}^{idle}$, as described in Section 3. To prove state-space model stability, we verify that all of the eigenvalues for the $N \times N$ matrix $A = [a_{mn}]_{M\times N}$ are less than 1.

The appropriate value for the number of phases $K$ is determined experimentally using our training data. To prevent overfitting, we randomly select 10,000 samples from
our training dataset of 21,480 points in order 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 4 show that four workload phases accurately capture the salient workload dependencies across all DVFS settings for both $K$-means and MLR phase classification. The accuracy is reported as $3\sigma$, where $\sigma$ 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 4 shows that the MLR approach yields superior estimation accuracy to $K$-means for all values of $K$.

Figure 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 counter metric in Figure 6 onto the first two principal components. Figure 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 and 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 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 8 and 9, respectively. We compare our phase-aware MLR and $K$-means modeling techniques to the MPC power modeling technique in Bartolini et al. [2011, 2012]. Using CPU power measurements in our training data, we estimate the parameters of $\tilde{g}(\ldots)$ in Equation (2) using nonlinear regression. By manually setting the processor voltage using the system BIOS, we establish the minimum voltage for each DVFS setting at which the system boots correctly, thus obtaining the operating voltages. We use the power estimated by Equation (2) multiplied by a proportional constant in the
state-space model with the same $a_{\text{run}}$ and $a_{\text{idle}}$ coefficients learned previously. Despite being able to estimate the total power consumption with an average absolute error of 0.62 W, the over-generality of Equation (2) leads to significant estimation error when performing thermal prediction. In addition to giving improved 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 thermal sensors and performance counters. The results show accuracy improvement for the proposed approach across the board, with an average 6.4% improvement in prediction accuracy relative to $K$-means and a 59% improvement relative to the MPC method.

6.2. 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 heterogeneous workload combinations. We evaluate each technique using a sequence of 15 combinations of between one and four 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 [Cochran and Reda 2010] to the following alternatives.

— *1-Phase.* This is the proposed approach but with a single workload phase. This corresponds to a single estimate for $\hat{g}(\cdot, \cdot)$ 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 phaseswitching technique [Coskun et al. 2008, 2009].
MPC. In this approach, we use the power model described by Equation (2) [Bartolini et al. 2011, 2012]. We incorporate it into a quadratic model prediction control (MPC) objective in place of the objective described in Section 5. With each control interval, the MPC controller predicts a sequence of DVFS settings that minimizes a quadratic cost function while maintaining the temperature within the thermal set point. The cost function is the squared deviation from the hard threshold with a prediction horizon of three control intervals (three seconds).

PI. We compare all modeling techniques to a simple proportional-integral PI feedback technique inspired by the reactive techniques used in commercial processors [Berktold and Tian 2010]. We calibrate the proportional and integral coefficients to the temperature-frequency gain observed for full processor utilization ($K_p = 0.12 \text{ GHz/}°\text{C}, K_i = 0.04 \text{ GHz/}°\text{C}$). The integral term is the sum of the soft-threshold deviations from the previous three control intervals.

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 I, 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 throughput 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 associated mechanical stresses, which translates into improved reliability. The MPC control objective, which uses the CPI power model in Equation (2), overestimates the temperature response to changes in the 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 5.8% improvement in instruction throughput over MPC with the same number of violations. The PI controller, while slightly less pessimistic, is calibrated to the case of maximum utilization. As 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 significantly reducing violations. While MLR and $K$-means produce comparable results, the improved accuracy of the MLR approach yields slightly fewer violations and higher performance.

In Figure 10, we show a comparison of frequency selections for the MLR, MPC, and PI approaches. The results show that 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.

7. CONCLUSION

In this work, 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. 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. 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 to 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 5.8% improvement in instruction throughput with the same number of thermal violations. In comparison to simple proportional-integral (PI) feedback control techniques, we improve instruction throughput by 3.9%, while significantly reducing the number of violations.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers who helped improve the content of this article.

REFERENCES


Thermal Prediction and Adaptive Control Through Workload Phase Detection


Received March 2012; revised July 2012; accepted October 2012.