Efficient Calibration of Thermal Models based on Application Behavior

Youngwoo Ahn, Inchoon Yeo and Riccardo Bettati
Texas A&M University
ayw0208@neo.tamu.edu, {ryanyeo, bettati}@cse.tamu.edu

***Abstract*— With increasing power densities, raising operating temperatures in chips threaten system reliability. Thermal control therefore has emerged as an important issue in system design and management.**

For dynamic thermal control to be effective, predictive thermal models of the system are needed. Such models typically use power as input, which renders them difficult to use in practical systems, where power monitoring is not available at processor or chip level. In this paper, we describe a methodology to infer the thermal model based on the monitoring of existing temperature sensors and of instruction counter registers. This allows the thermal model to be easily established, calibrated, and recalibrated at runtime to account for different thermal behavior due to either variations in fabrication or to varying environmental parameters. We validate the proposed methodology through a series of experiments. We also propose and validate an extension of the model and associated methodology for multicore processors.

I. Introduction

There has been a rise in interest over the recent years in thermal management to address reliability and performance of systems in a variety of settings, ranging from embedded systems to data center computing [3], [4], [5], [8], [9], [11], [12]. In general, thermal control and management can be realized at design time through either packaging or active heat dissipation, or at run time through various forms of *dynamic thermal management* (DTM). It has been shown that relying on thermal management through packaging or active heat dissipation means alone is expensive or impractical for high-performance processors [15]. As a result, a number of dynamic thermal management approaches have been proposed, ranging from dynamic voltage and frequency scaling (DVS) to clock gating and to architecture-level mechanisms that balance the load across functional units or cores on the processor chip. *Reactive* DTM triggers appropriate thermal control actions (e.g. DVS or clock gating) whenever readings from thermal sensors exceed a particular level [2], [3], [4], [8], [16]. *Proactive* DTM, on the other hand, allows for optimal thermal control by predicting thermal trajectories ahead of time as a function of CPU frequencies [13] and task executions [17]. While proactive DTM has been shown to allow for better utilization of computational resources compared to reactive schemes [13], it is also significantly more difficult to implement in practice. Most notably, its effectiveness relies on the accuracy of the *thermal model* that underlies the prediction of the effects of speed scaling and task execution on CPU temperature.

The thermal behavior of microelectronic circuits is typically modeled using variations of an RC circuit [7]. Examples of applications of such RC models can be found in [6] for web farms, in [13] for optimal speed control, and in [14] as part of the HotSpot thermal simulation.

A major impediment when putting proactive thermal management into practice is the need to develop a thermal model that is appropriate for the platform at hand. Generic thermal models only loosely capture the thermal behavior, due to variability in fabrication, environmental effects, or the need for special configurations of either cooling devices or other aspects of packaging. The models must therefore be appropriately calibrated before use. Figure 1 illustrates the effect of different cooling environments (high/low fan speeds vs. external fan) on the thermal trajectory. Note that the thermal increase rate and the highest temperature level varies for different types of cooling support even with the workload and the CPU remaining the same. When the thermal management must deal with such variabilities, effective configuration and efficient calibration methods are needed to acurately parameterize the thermal models that drive the predictive thermal control. Unfortunately, as thermal models describe the relation between input power and thermal behavior, they rely on the availability of a measurement of input power for parameterization purposes. Since such measurements are typically not available at processor level in most systems, we need to find alternative ways for thermal model calibration.

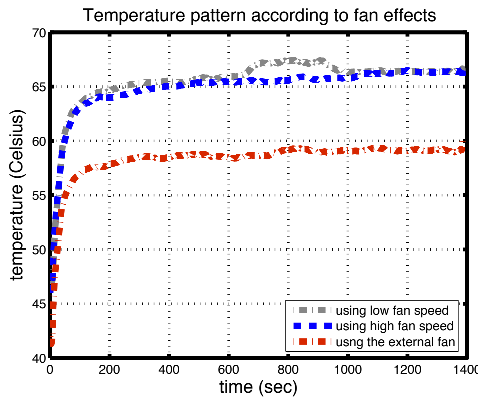


Fig. 1. Thermal trajectories with various cooling environments.

In this paper we describe an indirect methodology for parameterization of thermal models, which relies on the off-line analysis of the thermal behavior for a reference application, for which we measure both the detailed utilization behavior (through *PAPI* performance measure counters) and the thermal behavior (through thermal sensors on the chip). We determine and later exploit a linear relation between energy consumption and utilization level to calibrate the thermal model for new *target applications* by comparing relative utilization levels. This results in accurate thermal model parameters without the need for power measurements.

The paper is organized as follows: In Section II we give a brief overview of the thermal model used in this paper and of its RC circuit representation. In Section III we describe our indirect approach to estimate the parameters of the thermal model based on measurements of a reference application (`462.libquantum` from the SPEC CPU 2006 benchmark suite in our case). We describe how we measure and use relative utilization levels to calibrate the thermal model for new applications. Section IV describes how we expand both the thermal model and the calibration approach to multicore processors, and Section V validates our approach with experimental results. Finally, we conclude with a summary and an outlook in Section VI.

II. System Model

Effective prediction of the thermal trajectory depends first on a faithful thermal model and then on the accurate estimation and calibration of the specific parameters of the model for the system at hand. In this section we describe our thermal model and then proceed to elaborate on parameterization approaches that rely on power measurements. In the following sections we will describe methods to estimate model parameters at run time without the need to measure power.

A. Thermal Model

Thermal modeling of microprocessor circuits has traditionally applied variations of the simple Fourier's Law of heat conduction (e.g., [3], [7]). We assume that the environment has a fixed temperature, and that temperature is scaled so that the ambient temperature is zero. We define $T(t)$ and $P(t)$ as the temperature and the power input at time t, respectively. Fourier's Law is then expressed as follows:

$$T'(t) = \frac{P(t)}{C} - bT(t) \quad , \tag{1}$$

where the parameters R and C are the *thermal resistance* and *capacitance*, respectively, and capture the thermal characteristics of the processor chip under consideration. The term $b = 1/RC$ represents the *power dissipation rate* and is the inverse of thermal time constant of the system.

Simple thermal situations such as described in Equation (1) are often represented as RC circuits. Figure 2 shows the thermal RC circuit for a single-core processor, and it includes both the processor core and the packaging. We assume that the initial temperature is T_0, i.e., $T(t_0) = T_0$. If P is a constant power input level during a task execution

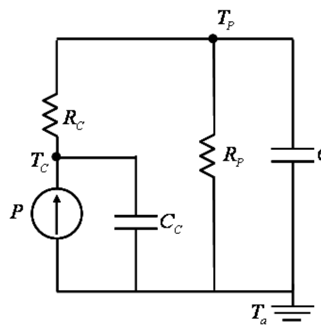


Fig. 2. Lumped RC Thermal Circuit Model for a Single-Core Processor

interval, then the core's temperature can be approximated by the thermal circuit model and Equation (1) as follows:

$$\begin{aligned} T_c(t) &= T_{ss.c}(1 - e^{-b_c(t-t_0)}) + (T_{c.0} - T_{p.0})e^{-b_c(t-t_0)} \\ &+ T_{ss.p}(1 - e^{-b_p(t-t_0)}) + T_{p.0}e^{-b_p(t-t_0)}, \end{aligned} \tag{2}$$

where $T_{ss.c} = R_c P$ and $T_{ss.p} = R_p P$ are steady-state temperatures of core and package respectively, b_p is the power dissipation rate of the package, and b_c is that of the core itself. The above approximation is derived from the fact that the dissipation rate b_c of the core is much larger than that of the package, b_p. This causes the core's temperature increase to be dominated by the core's own thermal characteristic immediately after the input power is first applied, and then the core's temperature becomes dependent on the package's thermal characteristic [13]. Hence, the temperature increases very quickly at the beginning and it slows down its increasing rate according to the increase of the package temperature.

III. Thermal Prediction based on Relative Utilization

Given a sampled thermal trajectory $T(t)$ of a system, the thermal parameters $T_{ss.c}$, $T_{ss.p}$, b_c, and b_p can be infered with help of Equation (2) if the input power P is known. In most systems, accurate power measurements are not available, and methods must be developed to capture the effect of input power onto the thermal behavior of the system without relying on absolute values for power levels. In the following we will describe how we use *relative utilization levels* to predict the thermal behavior of systems with the help of off-line measurement of thermal trajectories. In order to keep the following discussion simple, we limit ourselves to constant-speed processors, where the power consumption is application dependent. The results can be easily extended to the case of discrete processor's speed levels, where the power consumption is defined by the application and by the speed control module of the system.

The application of non-linear regression of Equation (2) on a measured thermal trajectory $T(t)$ yields estimations for $T_{ss.c}$, $b_c = 1/R_c C_c$, $T_{ss.p}$, and $b_p = 1/R_p C_p$. We note that the thermal model cannot be derived directly from the measurements, since *(a)* we do not know the input power P, and *(b)* we would have to factorize the results in order to get to the individual parameters. We therefore use an *indirect* approach - based on *relative utilizations* - to predict the thermal trajectory without knowledge of the input power level. We base our approach on the following observation:

Observation 1: Assume a system with a constant-speed processor, and two applications Γ_1 and Γ_2 that execute workload amounts X_1 and X_2 (measured in number of instructions) during a given interval $[0, t]$, respectively. Let E_1 and E_2 be the energy consumed by Γ_1 and Γ_2 during the same interval. The energy consumption ratio $\rho_e = E_1/E_2$, is equal to the utilization ratio $\rho_x = \frac{X_1/t}{X_2/t} = X_1/X_2$.

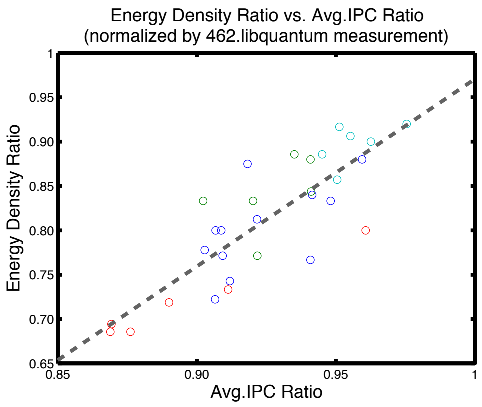


Fig. 3. Energy Density vs. Average IPC

The above observation is supported by experimental results. Figure 3 shows that the energy ratio during a given interval is proportional to the average utilization ratio for the same interval. The plot is derived from 30 sampled executions from 5 different benchmarks[1]. Li *et al.* make a similar observation about the relationship between the average power and the utilization (IPC) in [10]. The energy density (average power) was computed from Equation (2) and the dashed line shows a linear regression of the data.

The simple relation between utilization ratio and energy consumption ratio can be used to estimate the parameters $T_{ss.p}^{target}$ and $T_{ss.c}^{target}$ of a new application Γ_{target} as follows:

Calibration:

Step 1: Measure the thermal trajectory $T_{ref}(t)$ of the reference application Γ_{ref}. (In our case we use the SPEC CPU 2006 benchmark `462.libquantum` because of its constantly high utilization level.)

Step 2: Determine the parameters $T_{ss.c}^{ref}$, $T_{ss.p}^{ref}$, $b_{c.ref}$, and $b_{p.ref}$ from $T_{ref}(t)$, for example through nonlinear regression.

Step 3: Measure the average utilization level IPC_{ref} of the reference application Γ_{ref}.

Run-Time Estimation:

Step 4: Measure the utilization level IPC_{target} of the target application Γ_{target}.

[1]We measure the utilization level using the `PAPI_ipc` function of the $PAPI$ performance monitoring library. At 1-second intervals, we call `PAPI_ipc` to retrieve the current instruction and cycle counter, respectively. From these values we can determine a measure for the utilization of the processor.

Step 5: Predict the thermal trajectory $T_{target}(t)$ using Equation (2) with parameters $T_{ss.c}^{ref} \times \frac{IPC_{target}}{IPC_{ref}}$, $T_{ss.p}^{ref} \times \frac{IPC_{target}}{IPC_{ref}}$, $b_{c.ref}$, and $b_{p.ref}$.

(In the following we will use the notation $b_{c/p}$ to denote both $b_{c/p}$ and $b_{c.ref/p.ref}$.) We note that Step 1 to Step 3 form the calibration of the thermal prediction system and are performed off-line to determine a reference level. They may need to be performed when a system is deployed to account for environmental factors affecting cooling. For systems with multiple speed levels this calibration may need to be run once for each processor speed level. For systems with dynamic active heat dissipation mechanisms (e.g. fans with multiple speed levels) the calibration may need to be repeated as well to account for the multiple levels of thermal dissipation. Step 4 and Step 5 are executed at run time to exercise the model for the target application.

We will describe later how the accuracy can be further improved by using smaller discrete time intervals to better track application dynamics. The following equation or inequality shows how the temperature can be approximated in terms of energy consumption. That is,

$$\begin{aligned}
T(t) &= \frac{P}{C}\int_0^t e^{-b(t-x)}dx + T_0 e^{-bt} \qquad (3)\\
&\leq \frac{P}{C}\int_0^t dx + T_0 e^{-bt} \qquad (4)\\
&= \frac{P \cdot t}{C} + T_0 e^{-bt} = \frac{E(t)}{C} + T_0 e^{-bt}\\
\Longleftrightarrow \quad & T(t) \leq \frac{E(t)}{C} + T_0 e^{-bt} \quad , \qquad (5)
\end{aligned}$$

where Equation (3) is the solution of the Fourier Law in Equation (1) for a constant input power P. For the transition from Equation (3) to Equation (4) we take advantage of the fact that $\int_0^t e^{-b(t-x)}dx = 1/b(1-e^{-bt}) \leq 1/b \cdot bt$. As a result, the temperature can be approximated with help of the energy consumed during the time interval $[0, t]$, which we denote by $E(t)$. Similarly, we approximate the term $(1 - e^{-bt})$ in Equation (2) by the term bt, which allows for the following simplifications of the equation:

$$\begin{aligned}
T_c(t) &\leq \frac{E(t)}{C_c} + (T_{c.0} - T_{p.0})e^{-b_c t} + \frac{E(t)}{C_p} + T_{p.0}e^{-b_p t}\\
&= R_c b_c E(t) + (T_{c.0} - T_{p.0})e^{-b_c t}\\
&\quad + R_p b_p E(t) + T_{p.0}e^{-b_p t} \quad . \qquad (6)
\end{aligned}$$

Thus, if we discretize a task execution period with the interval of length Δ, the temperature variation is expressed by the energy consumption as follows in a recursive way,

$$\begin{aligned}
T_c((i+1)\Delta) \approx\ & R_c b_c E(\Delta) + (T_c(i\Delta) - T_p(i\Delta))e^{-b_c\Delta}\\
& + R_p b_p E(\Delta) + T_p(i\Delta)e^{-b_p\Delta}. \qquad (7)
\end{aligned}$$

We note that b_c and b_p have been previously determined, and therefore $e^{-b_{c/p}\Delta}$ are constants.

Given the above thermal approximation based on the energy consumption, we are able to estimate the thermal trajectory of any task execution as a result of monitored IPC values. The selection of the interval length Δ depends on the thermal parameters b_c and b_p so that the error between the estimated and the real temperature values be acceptably small. At the i^{th} interval, the temperature error by the energy approximation is described as follows,

$$\begin{aligned} error(i\Delta) &= R_c b_c E(\Delta) + R_p b_p E(\Delta) \\ &\quad + (T_c((i-1)\Delta) - T_p((i-1)\Delta))e^{-b_c\Delta} \\ &\quad + T_p((i-1)\Delta)e^{-b_p\Delta} \\ &\quad - R_c P(1-e^{-b_c i\Delta}) - R_p P(1-e^{-b_p i\Delta}) \\ &\leq R_c b_c P\Delta + R_p b_p P\Delta. \end{aligned} \tag{8}$$

Thus, suppose for example that $\Delta = \frac{1}{b_p 20} = \frac{1}{b_c 10}$, $R_c P = T_{ss.c} = 20$, and $R_p P = T_{ss.p} = 15$, the maximum error is $2.75^o C$. And the error decreases with decreasing Δ.

A. Temperature Predictions for Tasks with Dynamic Utilization

The temperature approximation method represented in Equation 7 can be easily extended to predict temperatures in systems where tasks display dynamic (i.e., time-varying) utilization levels. We exploit the relation between utilization level and energy consumption previously determined in Step 2 of the calibration. Given the estimated average energy consumption E_{ref} of the reference application, Equation 7 for the thermal trajectory can be modified as follows:

$$\begin{aligned} T_c((i+1)\Delta) &\approx R_c b_c E_{ref} \cdot \rho_{ipc,i} + (T_c(i\Delta) - T_p(i\Delta))e^{-b_c\Delta} \\ &\quad + R_p b_p E_{ref} \cdot \rho_{ipc,i} + T_p(i\Delta)e^{-b_p\Delta} \quad , \end{aligned} \tag{9}$$

where $\rho_{ipc,i} = \frac{IPC_{target}(i)}{IPC_{ref}}$ is the utilization ratio during interval i.

Although the selection of energy as a control factor and the local linearization of the model result in an overestimation in terms of the temperature increase, we will show in Section V that the error is acceptable for the thermal estimation by the appropriate choice of timing discretization length. In addition, by safely overestimating the thermal trajectory, this methodology can be applied in thermal management systems that require conservative handling of thermal behavior.

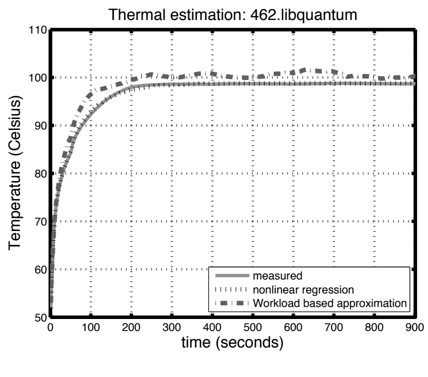


Fig. 4. Thermal parameter estimation & Utilization based temperature approximation.

IV. Extensions of Thermal Prediction Model for Multicore

In this section, we derive the estimation of thermal trajectories in dual-core system. For the multicore systems, the RC thermal circuit can be derived as shown in Figure 5. Each core's thermal trajectory can be expressed as follows,

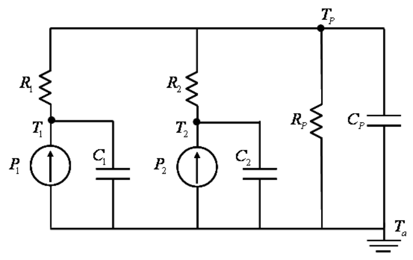


Fig. 5. A lumped RC Thermal Circuit Model for Dual Core Processors

$$\begin{aligned} T_1(t) &= R_p(P_1+P_2)(1-e^{-b_p t}) + T_{p.0}e^{-b_p t} \\ &\quad + R_1 P_1(1-e^{-b_1 t}) + (T_{1.0} - T_{p.0})e^{-b_1 t} \\ T_2(t) &= R_p(P_1+P_2)(1-e^{-b_p t}) + T_{p.0}e^{-b_p t} \\ &\quad + R_2 P_2(1-e^{-b_2 t}) + (T_{2.0} - T_{p.0})e^{-b_2 t}, \end{aligned} \tag{10}$$

where C_p, b_p, and $T_{p.0}$ are thermal parameters and the initial temperature of the packaging - for example the heat-sink contacts shared by both cores. Similarly to the single-core case, we estimate the parameters as described in Section III based on the thermal trajectory caused by a reference application (`462.libquantum` in our case) on each core. Given the symmetric arrangement of cores in many architectures, one may be tempted to use a system-level model for each core. However, due to variations in fabrication and external asymmetries caused by the layout and the integration of the system with external cooling mechanisms, each core is likely to have different thermal characteristics. It is advantageous, therefore, to derive the thermal parameters separately for each core. Similarly to Equation (6), for the single-core case, Equation (10) can be approximated in terms of energy consumption as well, i.e.,

$$\begin{aligned} T_1(t) &= R_1 b_1 E_1 + (T_{1.0} - T_{p.0})e^{-b_1 t} \\ &\quad + R_p b_p(E_1+E_2) + T_{p.0}e^{-b_p t} \\ T_2(t) &= R_2 b_2 E_2 + (T_{2.0} - T_{p.0})e^{-b_2 t} \\ &\quad + R_p b_p(E_1+E_2) + T_{p.0}e^{-b_p t}. \end{aligned} \tag{11}$$

We note that Equation(10) indicates how the thermal behavior of each core is affected by the other core's input power and the temperature level throughout the package.

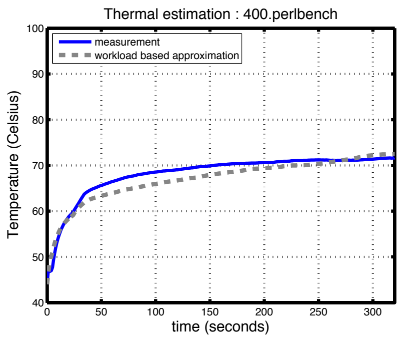
V. Results and Analysis

A. Experimental Setup

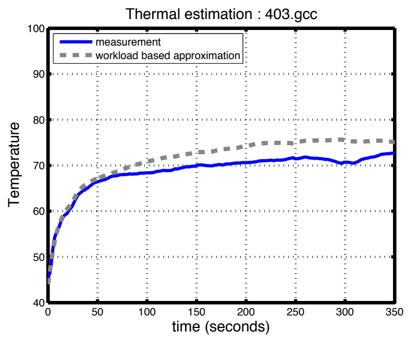
We evaluate the proposed thermal model in a real-world CMP product. In order to estimate each core's working temperature individually, we develop a specific device driver for accessing the Digital Thermal Sensor (DTS) in our multicore system at runtime. The trigger point of these thermal sensors is not programmable by software since it is set during the fabrication of the processor [1]. To validate our thermal model, we conduct our experiments using the system described in Table I.

B. Experimental Results

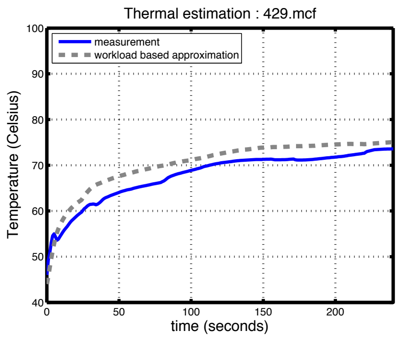
Figures 6(a)-6(f) show the measured temperatures and the comparisons with the predicted thermal trajectories. The results show how the approximations match the measured thermal trajectories.



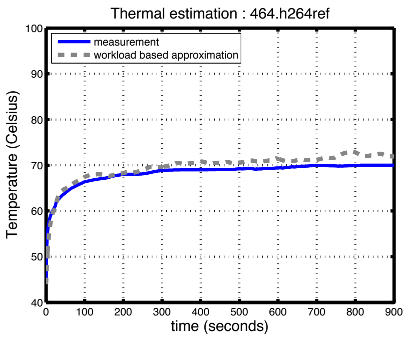
(a) 400.perlbench



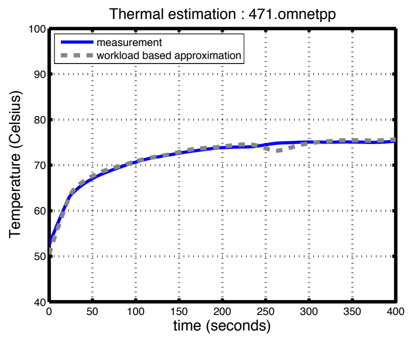
(b) 403.gcc



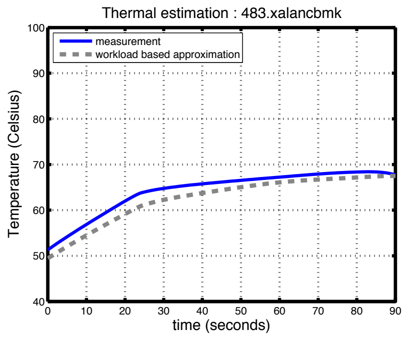
(c) 429.mcf



(d) 464.h264ref



(e) 471.omnetpp



(f) 483.xalancbmk

Fig. 6. The temperature estimation in Intel Quad-Core Q9650 processor.

TABLE I

EXPERIMENTAL SYSTEMS DESCRIPTION

	System
The number of cores	4 cores
Processor	Intel Quad Core Q6600
Memory Size	1 GB
Operating System.	SUSE 10.3 (Kernel Version: 2.6.27)

We define the error as:

$$error = \sqrt{\frac{\sum_{i=0}^{(t_f/\Delta)} \mid T_m(i\Delta) - T_a(i\Delta) \mid^2}{(t_f/\Delta)}}, \qquad (12)$$

where T_m is the measured temperature, T_a is the estimated value, and t_f is the measurement interval length in seconds. Table II shows the average error, which is found to be at most 3.5^oC for these benchmarks. Note that we should be able to decrease the error when a shorter monitoring time interval is selected for IPC.

TABLE II

AVERAGE ERROR OF UTILIZATION BASED THERMAL ESTIMATION

Benchmark	Avg.Error (^oC)
400.perlbench	2.09
403.gcc	3.47
429.mcf	3.49
464.h264ref	1.73
471.omnetpp	0.98
483.xalancbmk	2.74

Figure 7 shows the experimental result for the dual-core case. In the experiment, two different benchmarks, `464.h264ref` and `462.libquantum` were executed on *core 1* and *core 2* concurrently. By the measured IPCs from each processor core, we estimated the thermal trajectory of each core based on the Equation (11) and compared the

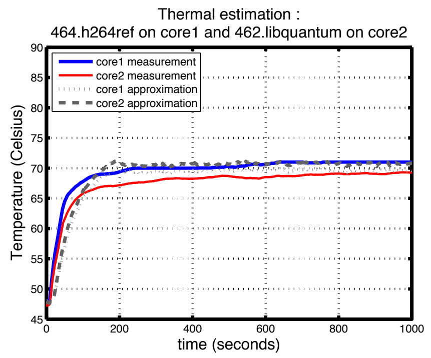


Fig. 7. The temperature estimation of two cores: 464.perlbench & 462.libquantum.

results with measured values.

VI. Conclusion and Future Work

Many of today's commercial computing systems and embedded devices do not have support for monitoring of power and energy. While this is rarely a problem for energy awareness (when in doubt, reduce speed), it becomes critical when attempting to predict the thermal behavior of a system. In this paper we develop a thermal model and a methodology for the efficient calibration of its parameters that does not rely on explicit information about input power. Rather, we closely monitor the thermal behavior of a well-known reference application on the given computing platform, and we then take advantage of the tight relationship between processor utilization and power consumption to predict the thermal trajectory for an unknown application at run time. The benefits of this methodology are numerous: First, the thermal characteristic of the computing platform can be explored after deployment time. For example, the system may run the reference application during configuration or even boot time and derive the thermal parameters shortly before run time. Next, it is easy to run this step for different levels of active heat dissipation if so desired. For example, the system may have a combination of on-package fans and case fans, which may be operated independently of each other. For each combination of such operating fans, the thermal characteristics of the system can be captured by the model and the parameters derived separately, thus enabling the effective prediction of thermal trajectories with dynamic active heat dissipation mechanisms. Finally, utilization information is readily available at run time, for example through Instructions Per Cycle (IPC) counter in PAPI.

In order to verify our thermal prediction model, we experiment with other benchmark applications for the prediction of temperature variations based on the IPC data monitored at every second. The experimental results show that the predictions are very close to the measurement of actual temperature values measured via Digital Thermal Sensor (DTS) embedded in each core despite the overestimation error that results from the energy-based approximation.

In the future work, we plan to extend the model to address IO-intensive applications, by better monitoring both reference and run-time application. This requires better information to be available about the proportion of computation-intensive instructions vs. total number of instructions at every monitoring interval. Similarly, the model must be extended to better reflect the thermal reality on the chip. In addition to IPC, other system information, like cache-misses or memory accesses, need to be considered for general applications.

References

[1] "Intel 64 and IA-32 Architectures Software Developer's Manual," http://support.intel.com/design/processor/manuals/.

[2] Y. Ahn and R. Bettati, "Transient overclocking for aperiodic task execution in hard real-time systems," in *Euromicro Conference on Real-Time Systems (ECRTS)*, July 2008, pp. 102–111.

[3] N. Bansal and K. Pruhs, "Speed Scaling to Manage Energy and Temperature," *Journal of ACM*, vol. 54, no. 1, pp. 1–39, 2007.

[4] D. Brooks and M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors," in *International Symposium on High-Performance Computer Architecture (HPCA-7)*, 2001, pp. 171–182.

[5] J. Choi, C.-Y. Cher, H. Franke, H. Hamann, A. Weger, and P. Bose, "Thermal-aware Task Scheduling at the System Software Level," in *International Symposium on Low Power Electronics and Design (ISLPED)*, 2007, pp. 213–218.

[6] A. Ferreira, D. Mosse, and J. Oh, "Thermal faults modeling using a rc model with an application to web farms," in *Proceedings of the 19th Euromicro Conference on Real-Time Systems (ECRTS 07)*, Pisa, ITALY, 2007.

[7] J.E.Sergent and A.Krum, *Thermal Management Handbook*. McGraw-Hill, 1998.

[8] 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 (DAC)*, 2006, pp. 548–553.

[9] J. Li and J. F. Martinez, "Power-Performance Implications of Thread-level Parallelism on Chip Multiprocessors," in *ISPASS*, 2005, pp. 124–134.

[10] T. Li and L. K. John, "Run-time modeling and estimation of operating system power consumption," *SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 1, pp. 160–171, 2003.

[11] P. Michaud, A. Seznec, D. Fetis, Y. Sazeides, and T. Constantinou, "A Study of Thread Migration in Temperature-Constrained Multicores," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 4, no. 2, 2007.

[12] F. Mulas, M. Buttu, M. Pittau, S. Carta, D. Atienza, A. Acquaviva, L. Benini, and G. D. Micheli, "Thermal Balancing Policy for Streaming Computing on Multiprocessor Architectures," in *Design, Automation, and Test in Europe Conference (DATE)*, 2008, pp. 734–739.

[13] R. Rao and S. Vrudhula, "Performance optimal processor throttling under thermal constraints," in *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, 2007, pp. 257–266.

[14] M. R. Stan, K. Skadron, M. Barcella, W. Huang, K. Sankaranarayanan, and S. Velusamy, "HotSpot: a Dynamic Compact Thermal Model at the Processor Architecture Level," *Microelectronics Journal: Circuit and Systems*, vol. 1, no. 1, 2003.

[15] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Baez, "Reducing power in high-performance microprocessors," in *Design Automation Conference*, 1998, pp. 732–737. [Online]. Available: citeseer.ist.psu.edu/375008.html

[16] S. Wang and R. Bettati, "Reactive Speed Control in Temperature-Constrained Real-Time Systems," in *Euromicro Conference on Real-Time Systems (ECRTS)*, 2006, pp. 73–95.

[17] J. Yang, X. Zhou, M. Chrobak, Y. Zhang, and L. Jin, "Dynamic Thermal Management through Task Scheduling," in *ISPASS*, 2008, pp. 191–201.