Optimizing Thermal Sensor Allocation for Microprocessors

Seda Ogrenci Memik, *Senior Member, IEEE*, Rajarshi Mukherjee, *Member, IEEE*, Min Ni, and Jieyi Long, *Student Member, IEEE*

***Abstract*—High-performance microprocessor families employ dynamic-thermal-management techniques to cope with the increasing thermal stress resulting from peaking power densities. These techniques operate on feedback generated from on-die thermal sensors. The allocation and the placement of thermal-sensing elements directly impact the effectiveness of the dynamic management mechanisms. In this paper, we propose systematic techniques for determining the optimal locations for thermal sensors to provide high-fidelity thermal monitoring of a complex microprocessor system. Our strategies can be divided into two main categories: uniform sensor allocation and nonuniform sensor allocation. In the uniform approach, the sensors are placed on a regular grid. The nonuniform allocation identifies an optimal physical location for each sensor such that the sensor's attraction toward steep thermal gradients is maximized, which can result in uneven concentrations of sensors on different locations of the chip. We also present a hybrid algorithm that shows the tradeoffs associated with number of sensors and expected accuracy. Our experimental results show that our uniform approach using interpolation can detect the chip temperature with a maximum error of 5.47 °C and an average maximum error of 1.05 °C. On the other hand, our nonuniform strategy is able to create a sensor distribution for a given microprocessor architecture, providing thermal measurements with a maximum error of 3.18 °C and an average maximum error of 1.63 °C across a wide set of applications.**

***Index Terms*—Allocation, dynamic thermal management (DTM), sensor, temperature.**

I. Introduction

TRENDS pertaining the power consumption and power densities on microprocessors are alarming. Number of devices per unit area as well as clock frequencies are increasing steadily. Although supply voltage levels and effective switched capacitance values are decreasing along with scaling, the rate of increase in the total number of devices on a chip largely surpasses this by orders of magnitude. For Intel microprocessors, the observation is that for every 1% increase in performance, there is a 3% increase in power consumption [1]. Increasing leakage power due to scaling is one important factor contributing to this phenomenon. The net result of these trends is that the microprocessor core power densities are alarmingly high [2], and extrapolations into the next decade indicate that this trend will uphold [1].

Power density directly affects the thermal profile of a chip. Thermal effects play an important role in various aspects of quality, correctness, and reliability of a system. Effective assessment and analysis of the thermal behavior of microprocessors is crucial to prevent the adverse impacts of thermal effects. A major challenge is the fact that the thermal behavior is input dependent and also sensitive to environment conditions. In addition, phenomena, such as process variation, affect the leakage power. This, in turn, affects the total power and the power densities, creating a coupling between power and thermal profiles. Thus, a highly accurate thermal profile of a complex system can only be established after it is deployed.

A direct means to capture the runtime thermal profile of a system is to utilize on-die thermal sensors. In this paper, we present a systematic methodology to allocate and place thermal sensors to build a thermal monitoring infrastructure for a complex microprocessor chip. Such infrastructures are used in modern processor architectures to assist dynamic-thermal-management (DTM) mechanisms. For instance, Intel Pentium 4, Pentium M, and IBM PowerPC processors are equipped with thermal sensors that trigger alerts if the junction temperature exceeds a specified limit. Based on these alerts, the processor power consumption is regulated via clock throttling [24]. While early solutions consisted of a few sensors per chip, the amount of sensors deployed on microprocessor tends to elevate in each generation. For example, IBM's POWER5 employs 24 digital temperature sensors [8]. In emerging multicore architectures, the total number of on-die sensors deployed will exceed far beyond. As a result, the decisions for allocation and placement of these sensors in the strategic locations on the chip need to be made within an automated framework considering the coverage accuracy and overheads involved.

Accuracy is crucial for thermal monitoring. Overestimation of temperature impacts performance negatively due to unnecessary triggering of thermal control mechanisms, e.g., dynamic voltage and frequency scaling [9]. On the other hand, Srinivasan *et al.* [29] have shown that the mean time to failure decreases exponentially with an increase in temperature. Therefore, underestimation of the die temperature is not desired since the processor will continue to operate at a higher temperature than its rated operating condition, hence greatly reducing the reliability.

One naive option to increase the accuracy is to place a very large number of sensors on the die. In current microprocessors,

Manuscript received June 20, 2007; revised August 8, 2007. This work was supported in part by the National Science Foundation (NSF) under Grant CNS-0546305 and in part by the NSF under Grant CCF-0541337. This paper was recommended by Associate Editor N. Chang.

S. O. Memik, M. Ni, and J. Long are with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208 USA (e-mail: seda@eecs.northwestern.edu; m-ni@northwestern.edu; jlo198@eecs.northwestern.edu).

R. Mukherjee is with the Synopsys Inc., Mountain View, CA 94043 USA (e-mail: rajmukherje@gmail.com).

Digital Object Identifier 10.1109/TCAD.2008.915538

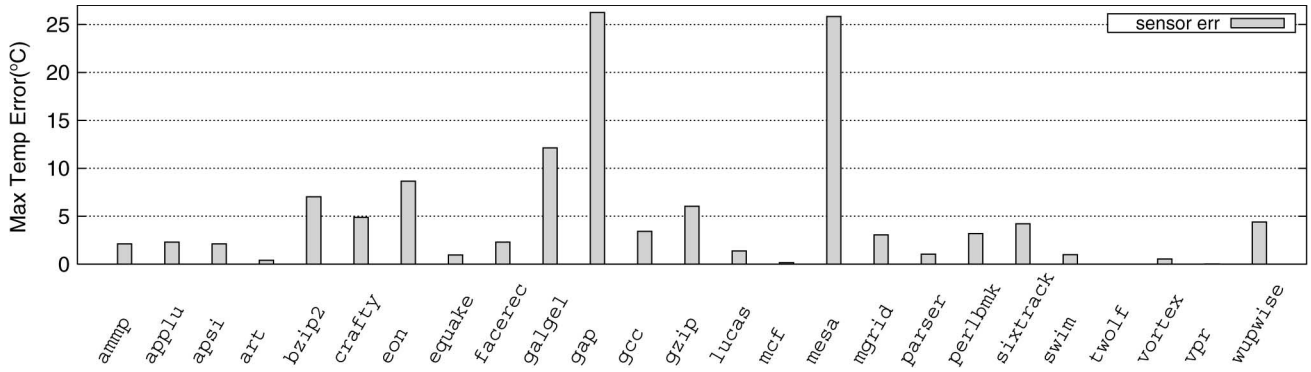


Fig. 1. Sensor reading errors of the SPEC2000 benchmarks if one sensor is placed at the hotspot indicated by a single application (twolf has been used in this example).

two types of sensors are being utilized. In the Intel Pentium 4 processor, the processor temperature is determined through an analog thermal sensor circuit consisting of a temperature-sensing diode, a factory-calibrated reference current source, and a current comparator [12]. One disadvantage of this sensor is that the threshold current (i.e., the current of the diode at the maximal allowable temperature) depends on some process parameters such as doping density; thus, each sensor should be calibrated individually. In fact, each processor is individually calibrated during manufacturing to eliminate any potential manufacturing variations [12]. This is a lengthy process and adds significant overhead to the testing and calibration phase. Sensors in the second category are more sophisticated in terms of design. They outperform the first kind of sensors in that their output signals can carry more information to assist the thermal management. Many new generation processors employ multiple on-chip digital thermal sensors of this type (such as IBM's POWER5). Each digital thermal sensor of POWER5 [8] consists of a ring oscillator whose frequency is controlled by a temperature-sensitive current reference and a counter that records the number of oscillations within a set time interval. Programmable registers define the maximum allowed temperature on each sensor. These sensors usually include a serial interface, such as I2C, SPI, or SMBus, which provides communication with embedded microcontrollers and other digital systems. Additional circuitry, such as analog-to-digital converter, is also needed to digitize the analog signals. The accuracy and the stability of digital sensors can be enhanced by increasing their sizes, which will further emphasize the constraints on hardware resources. In addition, as mentioned earlier, a calibration of each chip and each individual sensor is still needed. Furthermore, allocating arbitrarily large number of such sensors will not only create a significant area overhead, but routing the data from the sensor registers to a central processing unit will also pose a challenge.

In this paper, we propose two different sensor-allocation strategies to automate the design of a thermal monitoring infrastructure. Our goal is to define an allocation of thermal sensors and their physical locations using systematic techniques. In this process, we aim to maximize the accuracy of the readings obtained from the sensor placement while bounding the associated overheads. One of our proposed methods is the uniform approach that places the sensors on a set of predetermined static grid. We present techniques to optimize the granularity of such a grid, hence the number of sensors used. The second strategy is the nonuniform approach, in which the sensors can be inserted at any location on the chip. This approach receives hints from the expected temperature profiles. In essence, the two alternative strategies provide solutions for two cases: 1) a workload-independent static sensor allocation and 2) an allocation that is tailored for a given expected workload.

In either approach, we further investigate tradeoffs between the complexity of the infrastructure versus the accuracy of the readings. In the case of the uniform approach, we evaluated interpolation methods that allow the virtualization of sensors by utilizing readings from a smaller set of physical sensors. In the context of nonuniform thermal sensor allocation, we present results after clustering sets of hotspots under the constraints of the number of sensors and the desired accuracy.

The remainder of this paper is organized as follows. In Section II, we discuss our motivation of exploring systematic strategies for sensor allocation. We present an overview of related work in Section III. In Section IV, we propose the uniform sensor-allocation strategy. The nonuniform approaches by thermal-aware k-means clustering algorithm are presented in Section V. Section VI presents our experimental methodology and results. We conclude with important results and a summary in Section VII.

II. Motivation and Overview

An ideal method to allocate a sensor on a microprocessor is to find the hottest region over a set of well-defined applications and place one sensor there. This would require a minimum number of thermal sensors to monitor the highest temperatures on the microprocessor. Early attempts to create a dynamic monitoring for microprocessors employed a similar rationale. For example, for the Intel Pentium 4 processor [15], one sensor is placed near the rapid integer arithmetic logic unit (ALU), which has been determined to undergo the most severe thermal stress. For instance, our experimental results show that placing sensors at the hottest locations determined for one application can cause large temperature errors for other applications, as shown in Fig. 1. If we identify the highest temperature observed on the core by performing thermal simulation of one benchmark and place one sensor in the center of the block exhibiting this

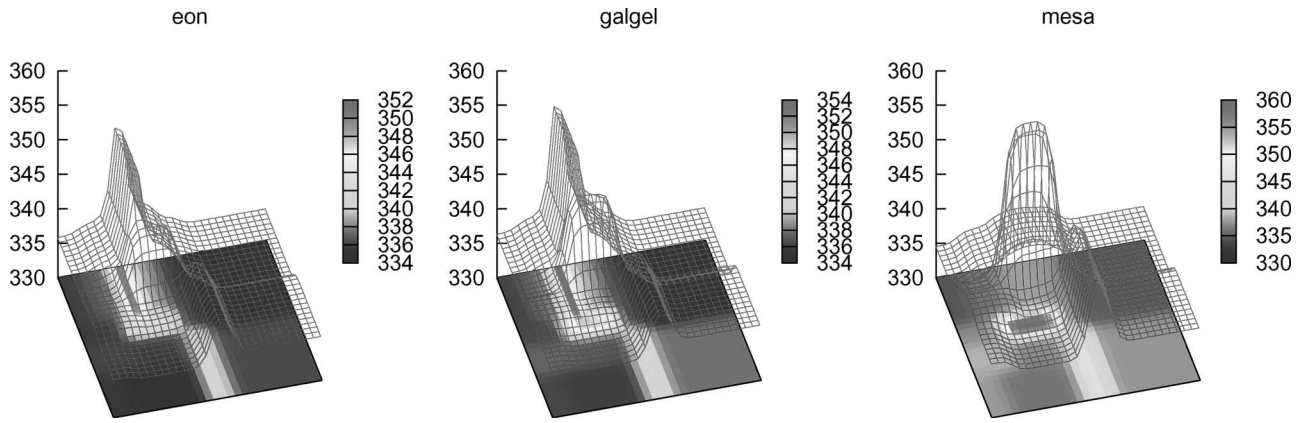


Fig. 2. Thermal profiles of the Alpha 21364 microprocessor with three different benchmarks from SPEC2000. It can be observed that although the hottest region is occurring near the IntQ, the locations are not the same (mesa has a clearly different hottest region). The location of the IntQ and other components is shown in the floorplan of Fig. 10.

highest temperature, then this decision can be highly unreliable. When we measure the amount of error resulting from relying on this sensor for all benchmarks, it can be as high as 27.4 °C.

The thermal behavior of microprocessors is affected by various factors. For example, localized heating on a processor is application dependent, as illustrated in the earlier example. In addition, process variations impact the total power consumption (by largely affecting the leakage component) and, hence, the temperature behavior of each chip, generating different thermal profiles. Power management techniques, such as local clock gating, further create a disparity in power densities among different regions on a chip.

It is intriguing to observe that recent studies aiming to identify the hottest regions of microprocessors reached diverse conclusions. The single thermal sensor on the Intel Pentium 4 processor is placed near the rapid integer ALU, which was identified as the likeliest candidate to cause a hotspot [15]. Skadron *et al.* [27] reported that in the Alpha 21364 architecture, the register file appears to be the hottest component consistently across a large set of SPEC CPU2000 [28] benchmarks. We performed yet another set of experiments with the same architecture (with a slightly different configuration, mainly using different configurations of the memory hierarchy, particularly the level 2 cache blocks; we experimented with a smaller sized L2-cache), benchmark suite, and thermal simulator. Our thermal analysis reveals that the issue queue (IntQ) generates the hottest points in most cases, as shown in Fig. 2. Considering the wide variety of sensor allocations in commercial products and previously reported results, it is safe to conclude that the temperature behavior is architecture and/or workload dependent. This motivates the need for a systematic approach to the sensor allocation and placement problem so that we can create the most effective monitoring infrastructure for a given architectural configuration, workload characteristics, and other relevant design and system-level parameters.

These observations lead us to two different directions to solve the thermal-sensor-allocation problem. If the thermal profiles over a large number of applications are not available, or a fully representative application set is not available, we need a workload-independent static approach. For instance, we can place a number of sensors on a regular grid, e.g., a 4×4 grid. We refer to this as the uniform thermal-sensor-allocation approach that does not rely on any temperature profiling data. The basic approach needs to be further improved in order to achieve high accuracy with a bounded size of the sensor grid. We developed virtualization techniques through the use of interpolation. The uniform sensor allocation can also monitor the localized temperature changes around the sensor's location.

On the other hand, if the thermal profiles simulated over a predictable and well-defined workload are available, we can opt to decide the locations of thermal sensors based on these data. The thermal profile of one application predicts one set of hotspots. A global hotspot map is the superposition of hotspots among all the simulated applications, as shown in Fig. 9. The problem of nonuniform sensor allocation is to identify n clusters among all these hotspots such that the total weighted reading error is minimized. We will elaborate on both approaches in Sections IV and V.

Furthermore, we identified two main purposes for thermal monitoring: global and local monitoring. The purpose of global monitoring is to track the hottest locations on the core to guide the DTM for thermal-emergency intervention. The local monitoring aims to establish a thermal monitoring for all core components even if some are not likely to exceed safe temperature thresholds. Although it is hardly possible for a usually cold component such as cache to trigger the thermal emergency, the highest temperature of such a component is still interesting for a thermal-induced leakage power control applied at the granularity of individual blocks. In addition, several local optimization techniques applied to various processor components have been proposed, including "heat and run" thread assignment for chip multiprocessors [23], activity migration to reduce hotspots [11], temperature-aware steering, clustering and thermal-aware renaming, and committing mechanisms [7]. All such efforts will require accurate local thermal monitoring for individual processor components to support fine-grain dynamic optimizations.

III. Related Work

While the design of accurate and efficient sensors has been studied extensively, only a few works have addressed the

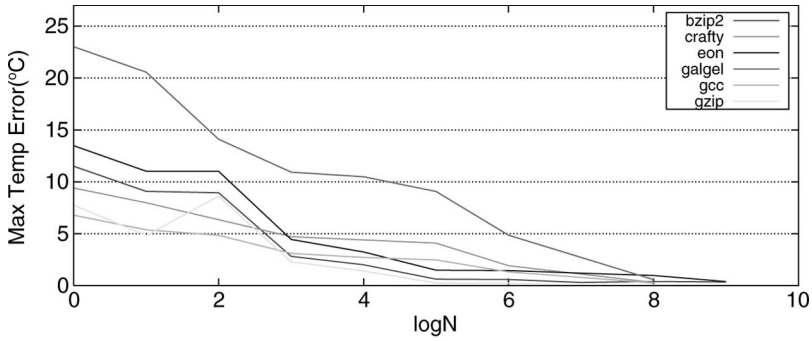


Fig. 3. Relationship between the number of sensors and the temperature-monitoring error. The x-axis is in log form.

sensor-allocation problem. Lee *et al.* [16] presented an analytical model that describes the relationship between the accuracy of a reading and the distance of the sensor from a hotspot. Gunther *et al.* [10] presented observations on thermal maps of a microprocessor and point to opportunities for optimized decisions on a sensor placement; however, they do not provide any solutions. Our previous work proposed the first systematic thermal-sensor-allocation scheme [21] for microprocessors. However, that work only considered nonuniform techniques and did not offer any placement methods where the thermal profiles are not available or dependable. In this paper, we have built upon this preliminary study and further introduced uniform placement strategies that do not depend on any *a priori* knowledge on workload or thermal profiling. We further create virtual sensors by interpolating among the uniformly placed sensors. We also made modifications to the original model in order to reduce the computational complexity required. These modifications also allow us to dynamically adapt a parameter of the model during actual temperature measurement, which was assumed to be a fixed constant in the original model. In this paper, we also presented an extensive set of results comparing different approaches—both our previous results as well as our new results on uniform placement.

On a different track, Bratek and Kos [4] presented a sensor placement for fault diagnosis of integrated circuits by linking temperature sensors and power modules in pairs. Lopez-Buedo *et al.* [17] investigated instantiating digital sensors on field-programmable gate arrays (FPGAs). Velusamy *et al.* [30] used such digital sensors to validate the accuracy of the thermal simulator HotSpot in modeling an FPGA-based system. Mukherjee *et al.* [22] proposed an algorithm to locate vacant configurable logic blocks that can be used to instantiate thermal sensors and embed them into a design mapped onto the FPGA. Thermal sensors are mainly used in FPGAs to validate the thermal behavior of a design against simulations during the prototyping phase. Their architectural constraints are fundamentally different than the microprocessors. While accuracy is the main goal for microprocessors while maintaining low overhead, in the case of FPGAs, the hardware cost of implementing a sensor can be much more dominant. Along the same lines, the sensor infrastructure is not intended to support the DTM, which is the focus for microprocessors. Therefore, the approaches and the priorities of optimization objectives are significantly distinct.

IV. Uniform Thermal Sensor Allocation

One straightforward method to create a workload-independent sensor infrastructure is to divide the chip into equally sized grids and place a sensor at each grid point. Then, all the sensors work in parallel, and the maximum temperature measured among the sensors will be used as the estimation of the core temperature.

If the grid size is equal to the effective sensing area of the thermal sensor, we can achieve 100% accuracy because each grid will have one sensor on it. However, it is usually not practical to implement such a fine-grain grid. Every sensor incurs additional cost, which becomes nonnegligible as the number of sensor increases. Early uses of thermal sensors relied on thermal diodes which are simple structures and easy to embed into any chip placement. However, the thermal diodes suffer from nonlinearity and can be subject to environmental effects [18]. New digital sensor designs are being proposed. The use of digital sensors on microprocessors, such as the POWER 5 from IBM, is becoming popular. Digital sensors consume nonnegligible resources (e.g., ring-oscillator-based counters, registers, bus interfaces, and routing to a central microcontroller) [3]. The placement of digital sensors into a highly optimized processor layout with a very limited white space will become complicated as the number of sensors increases. Moreover, collection and processing of the data generated by a vast number of sensors presents itself as a challenge. The limiting factor in most cases is the capability of routing the sensor readings to a processing unit. Such a solution will certainly have area, reliability, and power overhead.

Therefore, we need to maintain a bound on the static sensor grid size, and at the same time, we need metrics to evaluate the expected accuracy of such a static grid. We observed the following key aspects in the behavior of the static sensor-grid formation. First, the accuracy obtained from a static grid placement is related to the number of sensors used; however, it is not a linear relationship. In Fig. 3, we observe that when the number of sensor grids increases above 2^5 (note that the x-axis is in a log scale), the errors of sensor readings for a collection of applications do not change any further. We obtained the data shown in Fig. 3 by using the same experimental setup, as described in Section VI-A. We have performed a sensor placement using a static grid distributed across the entire core, and we have measured the maximum sensor reading error. Even if the reading error observed decreases to a substantially low plateau

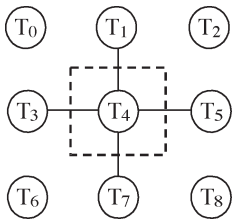


Fig. 4. Interpolation of the sensor data to obtain more accurate estimation of hotspots.

for some applications, for others, the error is still significant. Consider the galgel benchmark. Even with 64 sensors placed on a grid on the core, the errors can be as high as 5 °C for some applications. Considering that these readings will be used for the DTM and the thermal-emergency management, such incorrect readings can put the processor in danger or invoke unnecessary interruptions. The conclusion is that the size of the grid improves the effectiveness of the sensor infrastructure in many cases; however, in others, the hotspots may simply be located such that even a sizable grid of sensors will be incapable of capturing the locations of significant thermal events.

To minimize these errors, we have developed an interpolation scheme that uses the existing sensor readings obtained from the uniform placement and estimates the temperature of the points that lie between sensors. As we have described earlier, placing an arbitrary number of sensors on the chip is not the best solution in terms of the associated overheads. Therefore, it is beneficial to maintain an upperbound on the number of sensors that will feed data to the central control mechanism at a given instant. At the same time, the basic naive approach previously described may fail to perform reliably even if we dedicate a very high number of sensors. These observations motivated us to investigate more effective techniques for sensor allocation and placement. Our interpolation method is described in the following.

A. Interpolation-Based Virtualization of Sensors

The inconsistent accuracy and potentially large errors in the basic uniform grid are caused by the fact that we are not able to optimize the distances of the sensor points to the hotspots by such a static placement. Therefore, a corrective measure is needed to further refine the readings obtain from the static uniform placement. The basic idea is that several sensor readings can be used to interpolate the thermal behavior of locations farther away from the actual physical sensors. Therefore, instead of using the sensor reading directly as an indicator of the hotspot temperature, we calculate a more accurate value of the highest temperature on the chip by interpolating the sensor readings in the neighborhood of that hotspot.

We use an example to illustrate our idea. Consider the sensors shown in Fig. 4. Assume that the reading by the center sensor (T_4) is the highest. In this case, the hottest point in the region must be close to sensor T_4, if not exactly at T_4. In particular, the hottest point should be within the dashed square. The sides of this square extend exactly midway between T_4 and its neighbors in four directions. Intuitively, if the temperature at sensor T_3 is higher than the temperature at T_5, the x coordinate of the hottest point must be on the left of T_4. In the same way, we can determine whether the y coordinate of the hottest point is positive or negative by assuming that T_4 is the origin.

In the x dimension, the temperature of virtual sensor can be approximated as follows:

$$T_m = T_4 + \frac{1}{2}(T_3 - T_5). \tag{1}$$

While considering the possibility of $T_3 > T_5$, if we assume for a y dimension $T_1 > T_7$, we obtain our temperature interpolation as follows:

$$T_m = T_4 + \frac{1}{2} \cdot (|T_3 - T_5| + |T_1 - T_7|). \tag{2}$$

We will use (2) in our experimental section described in Section VI.

V. Nonuniform Sensor Allocation

Although we have so far analyzed the static grid-based locations for sensors, the sensors can also be placed at nonuniform locations on a chip if the thermal profiles of the microprocessor are available among a wide range of applications. One straightforward approach is to first detect the potential hotspots through simulation and then put a thermal sensor on each hotspot. This method has several problems in practice. First, the locations of hotspots are highly application dependent; hence, an optimal location for one application will not be the best solution for another. Second, the number of hotspots can be very large if we run a sufficiently large number of benchmarks to reveal the possible hottest locations on the chip.

Therefore, it is necessary to derive an automated scheme to decide the optimized allocation of a set of sensors. This procedure can be carried out in two stages. The first step is to generate a full thermal hotspot map across a wide range of applications. After that, the problem can be formulated as a clustering of the points of interest in the spatial domain. The number of clusters is decided by the number of available sensors. The center of each cluster will indicate the physical location of a sensor. This sensor will monitor the points associated with that cluster. Hence, the temperature reading from that sensor is representative of its respective coverage area. The reading error of each sensor can be represented by the distance between the cluster center and the other points in the cluster. Therefore, minimizing the total reading error is equivalent to solving a k-means clustering problem [19].

In the remaining part of this section, we will first briefly introduce the basic idea of the k-means clustering algorithm. Based on the basic k-means algorithm, we propose a modified thermal-aware k-means clustering method. Several different sensor-allocation strategies based on the thermal-aware k-means clustering are discussed in Section V-C.

A. Basic k-Means Clustering

The k-means clustering technique can be defined as follows [19].

Problem 1: Given an integer k and a set of n data points $R = \{\vec{a}_i | (x_{i1}, x_{i2}, \ldots, x_{im}),\ i = 1, 2, \ldots, n\}$ in an m-dimensional space, determine k centers such that the mean-square distance from each data point to its nearest center is minimized, i.e.,

$$\text{minimize} \quad \sum_{i=1}^{n} |\vec{a}_i - \vec{c}_i|^2 \tag{3}$$

where $\vec{c}_i$ represents the nearest cluster center to the data node $\vec{a}_i$.

The k-means clustering algorithm works by iteratively refining the position of the k cluster centers. Initially, the k cluster centers are randomly picked up from the n data points. Then, each data point finds out which center it is closest to. In this way, each cluster center will "own" a set of data points. The next step is to move the cluster center to the centroid of the points it owns. The iterations will continue until we reach the optimal solution.

The termination of the basic k-means algorithm is easy to prove. Each iteration will arrive at a new clustering configuration because the objective function value must be reduced. On the other hand, there are only a finite number of ways of partitioning the n data points into k groups. Therefore, the algorithm will terminate eventually.

However, it is not possible to guarantee the optimality of the basic k-means algorithm [20]. Therefore, some guidance must be applied to the initial random distribution. One effective heuristic introduced in [20] is to place jth initial center on the data points that are as far away as possible from the closest of center 1 through $j-1$.

For our purpose of the uniform sensor allocation, k sensors (k clusters) need to be created to monitor n hotspots. The cluster center corresponds to the sensor location. The data points in each cluster correspond to the hotspots that will be monitored by the sensor located at the cluster center.

B. Thermal-Gradient-Aware k-Means Clustering

The basic k-means algorithm correlates the sensor error with the distance between the hotspot and the location of sensor. This implies a linear relationship between the temperature gradient around a hotspot and the temperature at that location. Therefore, if we have, for example, one point with the highest temperature of 370 K and another point with the highest possible temperature of 350 K, the basic k-means algorithm will put the sensor in the middle between these two hotspots. However, it is beneficial in terms of accuracy if we move the sensor toward the 370-K point because the temperature gradient around a high-temperature location is larger than that at a low-temperature point. Therefore, a better sensor allocation should place an emphasis on minimizing the actual distance weighted by the temperature of the hotspots instead of using physical distances directly. Here, we propose a thermal-gradient-aware k-means clustering and a sensor-allocation algorithm to overcome this challenge.

Considering the thermal characteristic t, i.e., the temperature, of the hotspots, each such element can be regarded as distributed in a 3-D space and described by a tuple (x, y, t).

Algorithm k-means clustering()

Input : $hot_{x,y,t}$[N]: array of hotspots locations
$cluster_{x,y,t}$[K]: array of sensor locations
Output: member[N]: array of membership of hotspots

```
0  Initialize cluster[K] to be K points in hot[N]
1  WHILE δ/N > threshold
2    δ ← 0
3    FOR i=0 TO N-1
4      FOR j=0 TO K-1
5        d ← |hot_{x,y,t}[i]-cluster_{x,y,t}[j]|
6        IF d < d_min
7          d ← d_min
8          n ← j
9      IF member[i] ≠ n
10       δ ← δ + 1
11       member[i] ← n
12     newsize[n] ← newsize[n]+1
13     new_cluster_{x,y}[n] ← new_cluster_{x,y}[n]+hot_{x,y}[i]
       + α(new_cluster_{x,y}[n]-hot_{x,y}[i])
       ×(hot[i(t)]-new_cluster[n(t)]/newsize[n])
14     new_cluster_t[n] ← new_cluster_t[n]+hot_t[i]
15   FOR j=0 TO K-1
16     cluster_{x,y,t}[j] ← new_cluster_{x,y,t}[j]/newsize[j]
17     new_cluster_{x,y,t}[j] ← 0
18     newsize[j] ← 0
```

Fig. 5. Pseudocode for the thermal-gradient-aware k-means clustering algorithm.

By using this representation, our sensor-allocation algorithm operates in two stages. In the first stage, we group the hotspots into clusters where elements in the same cluster exhibit both spatial and thermal correlation. In the second stage, we identify the physical location within each such cluster where a thermal sensor should be placed. The sensor placed at this location would provide the most reliable information regarding the thermal condition of any hotspot within a certain cluster. If the sensor is placed directly at the center of the cluster, it is called a 3-D-placement scheme; otherwise, the location can be decided by some heuristic, as will be discussed later.

The clustering stage in a 3-D space is similar to the basic k-means iterations in a 2-D space, except that we need to extend the 2-D Euclidean distance to a 3-D case. That is, we need to use (4) to find the cluster center to which each hotspot belongs

$$d(i, j) = (x_i - x_j)^2 + (y_i - y_j)^2 + (t_i - t_j)^2 \tag{4}$$

where $(x_i,\ y_i)$, $(x_j,\ y_j)$, and $(t_i,\ t_j)$ are the coordinates and temperature of hotspot and cluster center, respectively.

Once we finish the hotspot clustering, we need to determine the physical location of thermal sensors. Instead of directly using the centroid of each cluster as the sensor location, we propose here another approach that takes into account the diversity of thermal gradients within a cluster. The basic idea behind this approach is to move the cluster centers or the sensors closer to the relatively higher temperature hotspots. This is equivalent to the sensor being attracted to the hotspots with high-temperature values with a larger force. The details are described in the algorithm shown in Fig. 5.

The 3-D Euclidean distance computation shown in (4) corresponds to Step 5) of the algorithm shown in Fig. 5. If the temperature of a certain hotspot is larger than the average of the

cluster, the sensor location will be pushed toward that hotspot by an attraction coefficient α. This is shown in Step 13). We have determined experimentally that an attraction coefficient value $\alpha = 0.1$ performs best. The cluster centers determined in Steps 3)–14) are then iteratively refined such that the mean-square distances of the hotspots from their respective cluster center are minimized over Steps 1)–18). Note that computing the cluster center using this method moves the sensor location physically closer to the steeper thermal gradients. We should also point to the fact that although we use the temperature dimension of the cluster centers, this dimension t is used only for modeling attraction toward points exhibiting high thermal stress. The temperature at the sensor's physical location determines its thermal reading, which has no physical relation with the temperature coordinate t of the cluster center.

Finally, we would like to point that the n-dimensional clustering concept allows us to consider other parameters during the sensor placement. In addition to the thermal gradients, frequency of occurrence of a certain hotspot may be another important parameter to guide the placement. We can incorporate this parameter as an additional dimension defining each point of thermal event point in the cluster sets.

Next, we will show how our proposed algorithm can be applied for different sensor-placement scenarios.

C. Sensor-Allocation Strategies

1) Global Sensor Allocation: In this strategy, the global hotspots are considered. We refer to the global hotspot as the absolute hottest point across the entire core. The global hotspots generally emerge in the same functional block repeatedly over many applications for a given architecture configuration (although the exact location of the hotspot may be shifted inside the block across applications). However, there can be reasons for the global hotspots to move into different components of the processor. For instance, in a superscalar processor, multiple copies of the floating-point unit can be selectively activated. During the intervals where a single floating-point unit is active, it can contain the hotspot. In another interval where multiple copies of this unit are active, the load would be distributed evenly, and the power density would be low in this location. At that point, a different unit, such as the instruction queue, can be the origin of the hotspot. Our strategy to place the sensors to capture such events works as follows. First, an initial number of sensors are estimated, and a thermal-gradient-aware sensor allocation is performed with that number. Then, the sensor allocation is changed iteratively until the results are within a given accuracy. A good starting point is to select the number of sensors to be equal to the total number of blocks and then to increase or decrease that number of sensors.

2) Local Sensor Allocation: In this case, our goal is to determine the allocation of the sensors for each individual processor component or block. We define a hotspot per component basis in this case. For a component, a hotspot is the location exhibiting the highest temperature within that block for an application. Effective local monitoring can be vital in various dynamic optimizations. Activity-migration and thread-assignment techniques [11], [23] can be assisted by local thermal-monitoring mechanisms. Temperature information regarding local components can be exploited by a dynamic cache optimization [13], [14] to reduce the leakage power. There can be different approaches for local sensor allocation.

Naive Allocation: The most straightforward approach is to place a fixed number of sensors per processor block. There are different ways to place the sensors based on the geometry and alignment of the block. The main idea is to recursively bisect the block into smaller units until the number of units is equal to the number of desired sensors. For example, this will involve placing a single sensor at the geometric center of the processor block. For two sensors, the block is bisected along the longest edge, and a sensor is placed at the center of each bisected rectangle.

Single Sensor at Thermal-Gradient-Aware WCC: This technique involves placing a single sensor for each processor component. This is equivalent to applying the thermal-gradient-aware allocation (TGA) shown in Fig. 5, without performing clustering within the component block. In this case, the entire hotspot map will form a single cluster, and the center of this cluster will be the sensor location for the single sensor. Hence, we term this as the thermal-gradient-aware weighted cluster center (WCC). It is the same as our thermal-gradient-aware k-means clustering approach but with k being equal to one.

TGA: In this approach, multiple sensors are placed within each processor block. Such allocation is performed by the thermal-gradient-aware k-means algorithm, where k is the number of sensors in each block, and the centers of the k clusters are the location for the k sensors. Increasing the number of sensors increases the monitoring accuracy. We observed that selecting the number of sensors to be two gives a good accuracy thermal sensing for microprocessor configuration in our experiments.

Hybrid Sensor Allocation: In contrast to the previous approaches where all blocks have an equal number of sensors (either single for the WCC strategy or a fixed number for the TGA strategy), we allow the allocation of variable number of sensors in different blocks. We adjust the amount of sensors necessary based on the error in measurement observed in different processor blocks. At first, for each processor block, a single sensor is allocated by the WCC method, and temperature-measurement error is determined. Then, depending on the amount of error observed in each individual block, the TGA is repeatedly applied with an increasing number of sensors until no significant improvement in accuracy can be observed in that block. By customizing the number of sensors required for accurate temperature measurement for each block, the total number of sensors is reduced.

VI. Experimental Results

In the following sections, we first describe our experimental methodology. In Section VI-B, we present our results for different sensor-allocation strategies.

A. Experimental Flow

We simulated the SPEC2000 benchmark (13 floating points and 12 integer benchmarks) suite [28] using SimpleScalar [6].

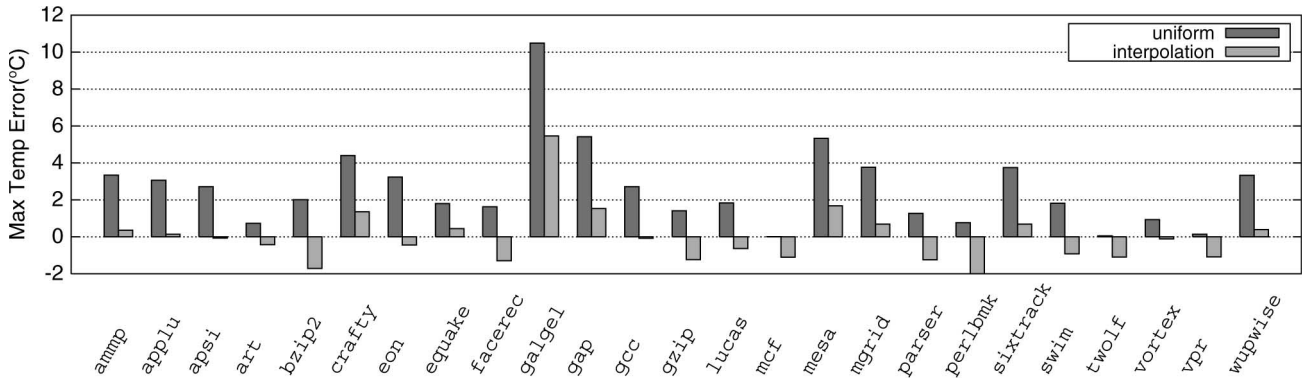


Fig. 6. Maximum temperature error for the SPEC2000 benchmarks performing a uniform global sensor allocation using the basic approach and the interpolation scheme.

The SimpleScalar simulates a superscalar processor with out-of-order issue and execution. We simulated each application for 200 million instructions after fast-forwarding an application-specific number of instructions, as proposed by Sherwood *et al.* [25]. We have chosen Alpha 21364 as our base processor. The Alpha 21364 processor is a four-way processor with a load store queue and a register update unit of sizes 32 and 64, respectively. The level 1 instruction and data caches are 64 KB and four way associative with a 32-B block size and a two-cycle latency. Unified level 2 caches are 512 KB and four way associative with 128-B line size and a uniform 15-cycle latency.

Wattch infrastructure [5] is used for architectural-level power modeling. The access patterns of the processor blocks from the SimpleScalar are then used by Wattch to compute the power dissipation of the blocks. The power data for 1.6 V at 1 GHz at 180-nm node were scaled using Wattch's linear scaling to obtain power for 1.3 V at 130-nm node with a 3-GHz clock frequency. We have used Hotspot version 3.1 [26] to perform the thermal simulation. The floorplan of Alpha 21364 and the power dissipation from Wattch are used as inputs to the Hotspot. The thermal simulation can be performed in the granularity of processor blocks. Another option is to perform a grid-level thermal simulation. In that case, the processor floorplan is uniformly divided into grids, and the temperature of each grid element is computed. The grid size determines the number of grid elements per processor component. Increasing the number of grid elements (higher resolution) helps capture the spatial variation in temperature per component. The initial temperature of the processor was assumed to be 60 °C. This represents the die temperature if the processor was already executing instructions prior to the execution of benchmarks to model the warm-up period. The ambient temperature is set to 40 °C. Power data at every 5 ms of simulation are used for transient thermal simulation at the grid level. For a 2.13-GHz clock frequency, this corresponds to a sampling period of every ten million cycles.

For our experiments, the point of interest is the relationship between the maximum temperature that would have been reported by the sensors within the core and the actual maximum temperature within that core. The expected temperature measurement of a sensor is determined by its location on the thermal map, i.e., the temperature of the grid location is equal to what a sensor placed in that grid location would have measured. We adjusted our grid size to ensure that each processor component contains a large number of distinctly monitored grid points, i.e., the granularity of the thermal simulation is much finer than the number of blocks on the processor floorplan. We also obtain the actual maximum temperature across all grid points on the core's floorplan directly from the thermal simulation. The difference between the maximum sensor temperature, i.e., the temperature at the grid point where the sensor has been placed, and the highest temperature across all grid points of the core floorplan indicates the "goodness" of our sensor-placement scheme. For the local sensor placement, the same arguments hold. In that case, however, we compare the maximum sensor reading and the maximum grid point temperature within a single processor component at a time.

For each application, transient thermal simulation is performed with a 5-ms sampling period, and we have performed 20 iterations for each configuration. The resulting instruction traces contain 200 MHz. For each iteration, we determine the maximum temperature on each core and the temperature reading that would be captured by the given sensor placement. The difference between these two temperatures yields an error value. Across all cores, we determine the largest error value. Across 20 iterations, we obtain 20 such maximum-error values. In all our results presented in the following section, the average error denotes the average of these 20 maximum-error values. Maximum error denotes the largest error value across these 20 individual maximum-error values.

B. Results

Our first set of results presents the maximum errors in sensor reading using the basic uniform approach on a 4×4 grid and the interpolation scheme, as shown in Fig. 6. The maximum error of the uniform approach among all the benchmarks is 10.48 °C, and the average error is 2.64 °C. Interpolation scheme improves these results significantly with a 5.47-°C maximum error and a 1.05-°C average error. In Fig. 6, the sensor error of interpolation method can be negative, which is not possible in any other approaches. As discussed before, overestimation would degrade the system performance by unnecessary triggering of the DTM. However, compared with the risk of putting the microprocessor in the thermal emergency, which degrades

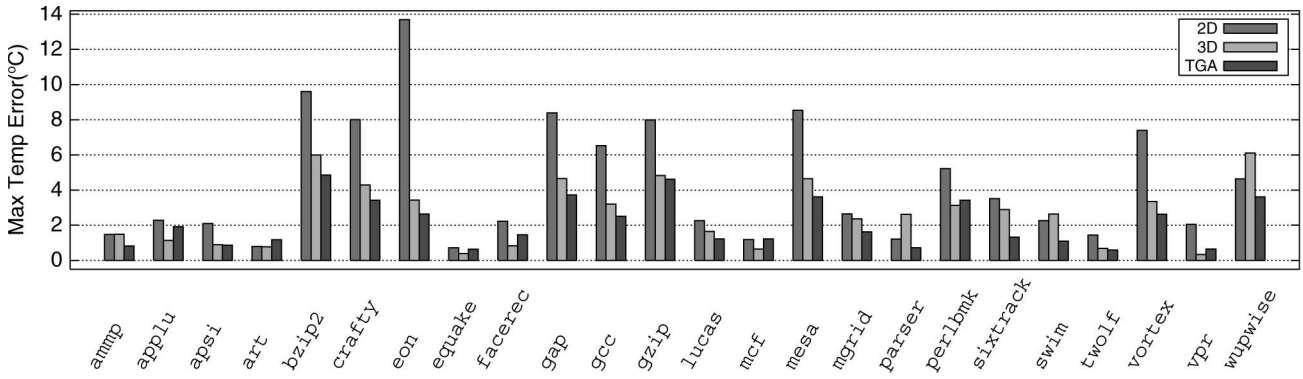


Fig. 7. Maximum temperature error for the SPEC2000 benchmarks performing a nonuniform global sensor allocation using 2-D allocation, 3-D allocation, and TGA approaches.

TABLE I
MAXIMUM ERRORS AND AVERAGE ERRORS FOR THE SPEC2000 BENCHMARKS USING DIFFERENT GLOBAL AND LOCAL THERMAL-SENSOR-ALLOCATION STRATEGIES IN A SINGLE-CORE MICROPROCESSOR

	Strategy	Max Err(°C)	Avg Err(°C)	Sensor #
Uniform Global	Basic	10.48	2.64	16
	Inter	5.47	1.05	16
NonUni Global	2D	13.69	4.58	16
	3D	6.11	2.66	16
	TGA	4.85	2.10	16
NonUni Local	Naive	22.96	10.79	36
	WCC	9.46	5.25	18
	2D	7.27	3.05	36
	3D	4.05	2.01	36
	TGA	3.18	1.63	36
	Hybrid	3.18	1.95	25

not only performance but also reliability, overestimation will be more acceptable.

The second set of results shown in Fig. 7 presents the maximum error in sensor reading using the nonuniform global allocation techniques described in Section V-C for allocating a total of 16 sensors across the entire floorplan. We observe that allocating a number of sensors equal to the number of blocks in the processor core could be a good starting point. Then, the number of sensors can be increased or decreased depending on the desired accuracy. Since the initial number of sensors is relatively low, a linear increase or decrease in the number of sensors to meet the accuracy proved to be a good method. For our experiments, we found that 16 sensors provide a good accuracy. We compared our method against three alternatives: 2-D allocation using only x and y coordinates, 3-D allocation using x, y, and t coordinates, and our thermal-gradient-aware approach. The results are summarized in Table I. It can be observed that the 2-D allocation can cause very big reading errors compared with the 3-D and thermal-gradient-aware approaches.

The third set of results shown in Fig. 8 presents the thermal monitoring accuracy for the local sensor-allocation techniques discussed in Section V-C. In our experiments, we observed that placing one to two sensors per block gives a good accuracy for our hotspot distribution. Of the different local allocation techniques, we present the thermal monitoring accuracy results for the WCC for a single sensor per block, the TGA approach for two sensors per block, and a hybrid assignment of one or two sensors for each block.

These sets of results are compared against all previously described techniques in Table I. This summary illustrates the most interesting trends for our proposed methods. For the hybrid method, initially, a single sensor is placed at the weighted centroid of each block, and the sensor errors are calculated. Then, we identified the blocks responsible for the maximum sensor error. We found that L2_left, FPAdd, FPReg, FPMul, FPMap, IntExec, and FPQ contributed the largest sensor errors. Note that out of the seven processor components, only L2_left belongs to the memory subsystem, and the rest are functional units. It can be observed from Fig. 9 that L2_left has two distinct hotspot clusters: one at the boundary of Icache and the other at FPMul. The WCC approach locates the sensor at the center of these two hotspots, and that is the reason for the large error. Using the thermal-gradient-aware method can improve the accuracy greatly if we place two sensors in each processor block.

Fig. 9 shows the sensor placement using the hybrid approach for local monitoring. The sensors are denoted by rectangles. This floorplan zooms onto the core of the processor. Dotted lines represent that the majority of the cache blocks have not been included in this floorplan. This figure shows the sensor placement for local monitoring. The hotspots depicted within each block represent the locations exhibiting the highest temperatures observed within the respective block across different benchmarks. Therefore, each block contributes to this hotspot map with its own local hotspot set, as shown in Fig. 9, and sensors have been placed to perform the local monitoring within each block.

For the memory subsystem, most of the hotspots occur at its boundary. This is due to the interaction of neighboring hot blocks on the relatively cooler ones. Other blocks, such as the floating-point multiplier and the integer execution unit, have higher power densities and higher maximum temperatures in general. For those blocks, we observe a set of hotspots in the interior regions and another set of hotspots at the boundaries, which occur due to the lateral thermal interaction with the neighboring blocks. It can be seen from the figure that two sensors were assigned to the blocks L2_left, FPAdd, FPReg,

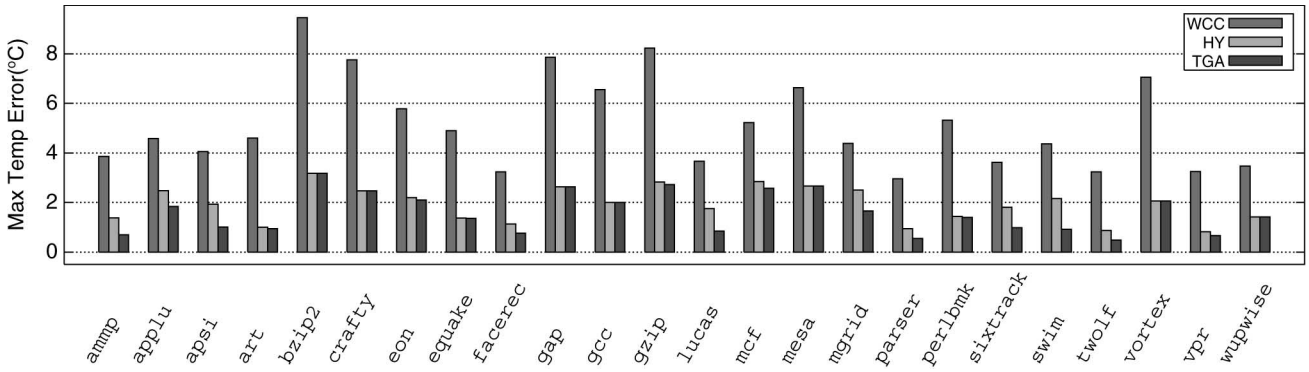


Fig. 8. Maximum temperature error for the SPEC2000 benchmarks performing a nonuniform local sensor allocation using WCC approach, hybrid approach, and TGA applied at the local level.

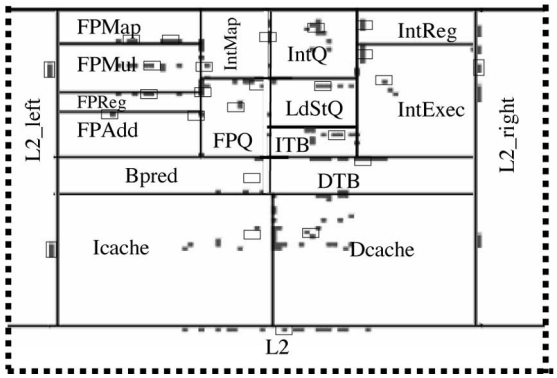


Fig. 9. Hotspot distribution across processor blocks and (marked as rectangles) the sensor locations determined by our hybrid-allocation strategy for local monitoring.

FPMul, FPMap, IntExec, and FPQ, and that single sensor was assigned to the rest.

So far, we have presented relative reading errors to assess the coverage success of the sensor infrastructures. In Fig. 10, we show the absolute temperature distribution for a representative application and the associated sensor readings generated by a global uniform sensor grid placement (4×4 array of sensors). We have obtained this set of results with extended sampling intervals and a larger total number of cycles. In this simulation, one sampling interval is composed of 125 million cycles, and 15 intervals have been sampled for a total of 1.875 billion cycles. This helps us also to demonstrate that the sensor behavior can be expected to be consistent for longer durations of execution. In addition, we would like to mention that our integrated simulation environment (SimpleScalar, Wattch, and HotSpot) has also been extended with a throttling mechanism that simulates the voltage and frequency scaling employed by microprocessors when a certain critical temperature has been exceeded. In this particular simulation, this threshold has been set to 82 °C.

C. Discussion

Table I summarizes the maximum error and the average of maximum errors for each of our sensor-allocation techniques. Our first observation is that our interpolation scheme improves the accuracy significantly over the basic uniform approach without increasing the number of sensors needed. Comparing the nonuniform techniques against each other, we observe once again that our proposed thermal-gradient-aware technique performs best. Comparing the uniform interpolation and nonuniform clustering-based techniques, we observe that the nonuniform technique and the interpolation-based uniform technique are comparable. The nonuniform placement can improve the error in a maximum temperature reading, which impacts the DTM decisions most. However, we note that this analysis is still based on one benchmark set. The nonuniform placement technique using clustering relies on profiling data and resulting thermal maps of hotspots. On the other hand, the uniform interpolation mechanism is built upon a static placement independent of such profiling assumptions. This provides us with two alternative approaches, where clustering yields the best quality in maximum temperature measurements if the profiling data are highly accurate and dependable; otherwise, the static placement with interpolation approach may be more suitable.

The uniform sensor allocation generally may not be preferred in the localized component-based temperature monitoring. The reason is that the necessary number of sensors depends heavily on the floorplan of the microprocessor. In some cases, this may require an arbitrarily high number of sensors. Our conclusions for the local monitoring using alternative nonuniform approaches are as follows. The thermal-gradient-aware technique outperforms all other clustering-based techniques that do not have sensitivity to the thermal gradients, using the same number of sensors. Our hybrid approach further builds upon the thermal-gradient-aware technique by selectively adjusting the number of sensors needed in each block. This leads to similar accuracy with fewer sensors used.

It can be noted that our scheme applies the best sensor allocation and placement decision that is possibly independent of the power and thermal model. For our experimental flow, we have used the Wattch infrastructure to obtain block-level power. It is true that the evaluation would be more accurate if we model power distribution inside individual components. However, such data require detailed component layouts, which are not public information. New versions of HotSpot, as used in our study, allow grid-based simulation efficiently. Our attempt to apply fine-grid thermal simulation on top of the blocks in the least gives better accuracy compared with

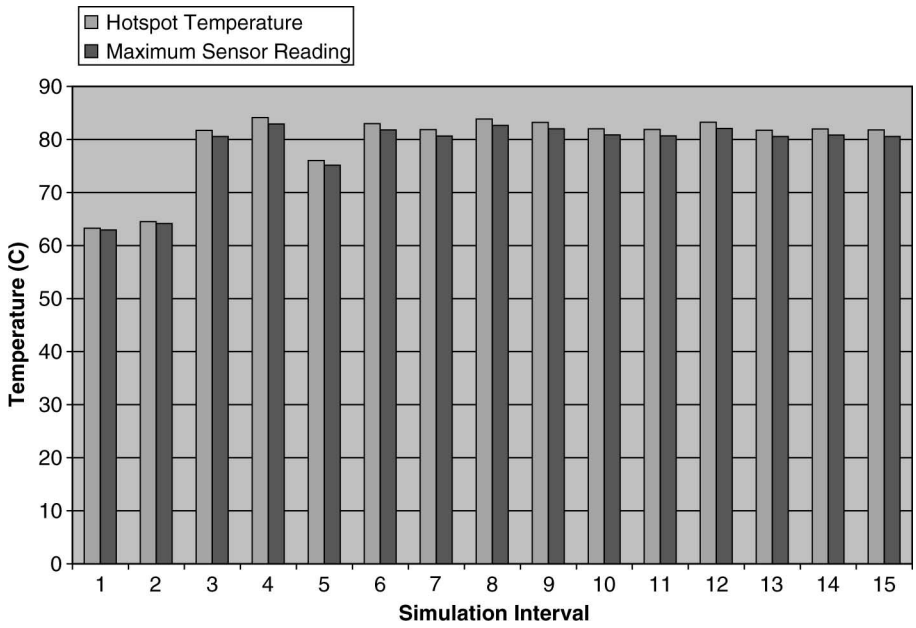


Fig. 10. Temperature values obtained from thermal simulation and the sensor readings for a uniform sensor grid (for the gzip benchmark).

performing block-based thermal simulation. The results for the evaluation of our technique correspond to the accuracy of the thermal simulator HotSpot. HotSpot has been validated against Floworks (http://www.floworks.com), which is a commercial finite-element simulator of 3-D fluid and heat flow [26].

Finally, regardless of whether a clustering-based method or a static scheme is used, some hotspots may still be missed. A drastically steep thermal gradient within a very small distance can lead to missing possible hotspots, even if we employ a static grid. Our interpolation scheme using the physical sensor readings to assess those locations falling in between physical sensors aims to remedy this problem. While there always may be some thermal gradients, which are challenging to monitor for any type of sensor network, with a reasonably sized static grid and the follow-up interpolation, most thermal gradients could be successfully monitored.

VII. Conclusion

We have presented techniques to generate a sensor infrastructure to monitor the maximum temperature on microprocessors. Our goal is to provide accurate temperature readings in a given system while maintaining a reasonable overhead in terms of the number of sensors and the sensor data collection effort. The DTM schemes can leverage on the sensor infrastructure that is built by our approaches.

We first analyzed the uniform sensor-allocation strategy, where the sensors are placed on a uniform grid on the microprocessor. This strategy can be applied in cases where the thermal profiles are not available or they vary greatly among different applications. We improved upon this naive solution by introducing an interpolation scheme. The interpolation is able to reduce the average errors for a given sensor distribution without incurring any additional overhead. Our experiments show that our uniform sensor-allocation strategy can provide a comparable accuracy, the nonuniform placement delivering a slightly better accuracy for maximum temperature monitoring, as that of the nonuniform strategy by the same number of thermal sensors. On the other hand, the nonuniform strategy can be used to monitor both the global and local thermal conditions. The uniform placement for local component-based monitoring can require a high number of sensors in total, and some processor components present much more predictable thermal profiles compared with the global case. Therefore, the nonuniform placement can be the most dependable and accurate option for such blocks. Our proposed thermal-gradient-aware approach reduces the maximum error to 4.85 °C and the average error to 2.10 °C in the global thermal monitoring. In the local thermal monitoring, the maximum error is 3.18 °C, and the average error is 1.63 °C.

References

[1] S. Borkar, P. Dubey, K. Kahn, D. Kuck, H. Mulder, S. Pawlowski, and J. Rattner, *Platform 2015: Intel Processor and Platform Evolution for the Next Decade*, 2005. Technology@Intel Magazine.

[2] S. Borkar, "Design challenges of technology scaling," *IEEE Micro*, vol. 19, no. 4, pp. 23–29, Jul./Aug. 1999.

[3] S. Bota, M. Rosales, J. Rosello, and J. Segura, "Smart temperature sensor for thermal testing of cell-based ICs," in *Proc. Des. Autom. Test Eur.*, 2005, pp. 464–465.

[4] P. Bratek and A. Kos, "Temperature sensors placement strategy for fault diagnosis in integrated circuits," in *Proc. Symp. Semicond. Thermal Meas. Manage.*, 2001, pp. 245–251.

[5] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proc. Int. Symp. Comput. Architecture*, 2000, pp. 83–94.

[6] D. C. Burger and T. M. Austin, "The simplescalar tool set, version 2.0," *Comput. Architecture News*, vol. 25, no. 3, pp. 13–25, 1997.

[7] P. Chaparro, G. Magklis, J. Gonzales, and A. Gonzales, "Distributing the frontend for temperature reduction," in *Proc. Int. Symp. High-Performance Comput. Architecture*, 2005, pp. 61–70.

[8] J. Clabes, J. Friedrich, M. Sweet, J. DiLullo, S. Chu, D. Plass, J. Dawson, P. Muench, L. Powell, M. Floyd, B. Sinharoy, M. Lee, M. Goulet, J. Wagoner, N. Schwartz, S. Runyon, G. Gorman, P. Restle, R. Kalla, J. McGill, and S. Dodson, "Design and implementation of the power5 microprocessor," in *Proc. Des. Autom. Conf.*, 2004, pp. 670–672.

[9] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," in *Proc. Int. Symp. Comput. Architecture*, Boston, MA, Jun. 2006, pp. 78–88.
[10] S. Gunther, F. Binns, D. M. Carmean, and J. C. Hall, "Managing the impact of increasing microprocessor power consumption," *Intel Technol. J.*, vol. 5, no. 1, pp. 1–9, Feb. 2001.
[11] S. Heo, K. Barr, and K. Asanovic, "Reducing power density through activity migration," in *Proc. Int. Symp. Low Power Electron. Des.*, 2003, pp. 217–222.
[12] Intel Inc. (2002, Dec.). *Intel Pentium 4 Processor With 512-KB L2 Cache on 0.13 Micron Process Thermal Design Guidelines*. [Online]. Available: http://download.intel.com/design/Pentium4/datashts/29864312.pdf
[13] J. K. John, J. S. Hu, and S. G. Ziavras, "Optimizing the thermal behavior of subarrayed data caches," in *Proc. Int. Conf. Comput. Des.*, 2005, pp. 625–630.
[14] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache decay: Exploiting generational behavior to reduce cache leakage power," in *Proc. Int. Symp. Comput. Architecture*, 2001, pp. 240–251.
[15] V. Krinitsin. (2005). *Pentium 4 and Athlon XP: Thermal Conditions*. [Online]. Available: http://www.digit-life.com/articles/pentium4athlonxp-thermalmanagement/
[16] K.-J. Lee, K. Skadron, and W. Huang, "Analytical model for sensor placement on microprocessors," in *Proc. Int. Conf. Comput. Des.*, 2005, pp. 24–27.
[17] S. Lopez-Buedo, J. Garrido, and E. I. Boemo, "Dynamically inserting, operating, and eliminating thermal sensors of FPGA-based systems," *IEEE Trans. Compon. Packag. Technol.*, vol. 25, no. 4, pp. 561–566, Dec. 2002.
[18] S. Lopez-Buedo, J. Garrido, and E. I. Boemo, "Thermal testing on reconfigurable computers," *IEEE Des. Test Comput.*, vol. 17, no. 1, pp. 84–91, Jan.–Mar. 2000.
[19] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.
[20] A. W. Moore. (2001). *K-means and Hierarchical Clustering*. [Online]. Available: http://www.cs.cmu.edu/awm/tutorials
[21] R. Mukherjee and S. O. Memik, "Systematic temperature sensor allocation and placement for microprocessors," in *Proc. Des. Autom. Conf.*, Jul. 2006, pp. 542–547.
[22] R. Mukherjee, S. Mondal, and S. O. Memik, "Thermal sensor allocation and placement for reconfigurable systems," in *Proc. Int. Conf. Comput.-Aided Des.*, 2006, pp. 437–442.
[23] M. D. Powell, M. Gomma, and T. N. Vijaykumar, "Heat-and-run: Leveraging SMT and CMP to manage power density through the operating system," in *Proc. Int. Conf. Architectural Support for Program. Lang. Operating Syst.*, 2004, pp. 260–270.
[24] E. Rotem, A. Naveh, M. Moffie, and A. Mendelson, "Analysis of thermal monitor features of the Intel Pentium m processor," in *Proc. Workshop Temperature-Aware Comput. Syst.*, 2004, pp. 29–35.
[25] T. Sherwood, E. Perelman, and B. Calder, "Basic block distribution analysis to find periodic behavior and simulation points in applications," in *Proc. Int. Conf. Parallel Architecture Compilation Techn.*, 2001, pp. 3–14.
[26] K. Skadron, K. Sankaranarayanan, S. Velusamy, D. Tarjan, M. R. Stan, and W. Huang, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Architecture Code Optimization*, vol. 1, no. 1, pp. 94–125, Mar. 2004.
[27] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *Proc. Int. Symp. Comput. Architecture*, 2003, pp. 2–13.
[28] SPEC-CPU2000, "Standard performance evaluation council, performance evaluation in the new millennium, version 1.1," *Electronic Resource*, 2000.
[29] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The case for lifetime reliability-aware microprocessors," in *Proc. Int. Symp. Comput. Architecture*, 2004, pp. 276–287.
[30] S. Velusamy, W. Huang, J. Lach, M. R. Stan, and K. Skadron, "Monitoring temperature in FPGA based SoCs," in *Proc. Int. Conf. Comput. Des.*, 2005, pp. 634–637.

Seda Ogrenci Memik (SM'05) received the B.S. degree in electrical and electronic engineering from Bogazici University, Istanbul, Turkey, and the Ph.D. degree in computer science from the University of California, Los Angeles.

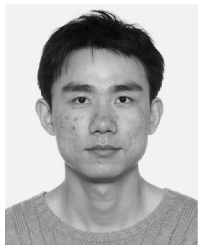
She is currently an Assistant Professor with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL. Her research interests include embedded and reconfigurable computing, thermal-aware design automation, and thermal management for high-performance microprocessor systems.

Dr. Memik has served as a Technical Program Committee Member, an Organizing Committee Member, and a Subcommittee Chair of several conferences, including International Conference on Computer-Aided Design, Design, Automation and Test in Europe, Field Programmable Logic and Applications, Great Lakes Symposium on Very Large Scale Integration (VLSI), and International Workshop on Applied Reconfigurable Computing. She is the Associate Editor of the IEEE TRANSACTIONS ONVLSI SYSTEMS. She was the recipient of the National Science Foundation Early Career Development Award in 2006.

Rajarshi Mukherjee (M'01) received the B.E.Tel.E. degree in electronics and telecommunication engineering (with honors) from Jadavpur University, Calcutta, India, in 2000, and the M.S. and Ph.D. degrees from the Northwestern University, Evanston, IL, in 2003 and 2006, respectively.

Before joining the graduate program at the Northwestern University, he was with the Texas Instruments, Bangalore, India, from 2000 to 2001. He is currently a Senior R&D Engineer with the Synopsys Inc., Mountain View, CA. His research interests include thermal-aware design and analysis techniques for integrated circuits and high-performance microprocessors, timing analysis, and high-level synthesis.

Dr. Mukherjee was the recipient of the Walter Murphy Fellowship Award from the Northwestern University in 2001–2002.



Min Ni received the B.S. and M.S. degrees in electronic engineering from Tsinghua University, Beijing, China, in 2002 and 2005, respectively. He is currently working toward the Ph.D. degree in computer engineering in the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL.

His research interests are on very large scale integration computer-aided design, particularly high-level and logic synthesis.

Jieyi Long (S'07) received the B.S. degree in microelectronics from Peking University, Beijing, China, in 2006. He is currently working toward the Ph.D. degree in the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL.

His research interests include thermal monitoring and management for high-performance very large scale integration systems.