Thermal Monitoring Mechanisms for Chip Multiprocessors

9

JIEYI LONG, SEDA OGRENCI MEMIK, and GOKHAN MEMIK
Northwestern University
and
RAJARSHI MUKHERJEE
Synopsys Inc.

With large-scale integration and increasing power densities, thermal management has become an important tool to maintain performance and reliability in modern process technologies. In the core of dynamic thermal management schemes lies accurate reading of on-die temperatures. Therefore, careful planning and embedding of thermal monitoring mechanisms into high-performance systems becomes crucial. In this paper, we propose three techniques to create sensor infrastructures for monitoring the maximum temperature on a multicore system. Initially, we extend a nonuniform sensor placement methodology proposed in the literature to handle chip multiprocessors (CMPs) and show its limitations. We then analyze a grid-based approach where the sensors are placed on a static grid covering each core and show that the sensor readings can differ from the actual maximum core temperature by as much as 12.6°C when using 16 sensors per core. Also, as large as 10.6% of the thermal emergencies are not captured using the same number of sensors. Based on this observation, we first develop an interpolation scheme, which estimates the maximum core temperature through interpolation of the readings collected at the static grid points. We show that the interpolation scheme improves the measurement accuracy and emergency coverage compared to grid-based placement when using the same number of sensors. Second, we present a dynamic scheme where only a subset of the sensor readings is collected to predict the maximum temperature of each core. Our results indicate that, we can reduce the number of active sensors by as much as 50%, while maintaining similar measurement accuracy and emergency coverage compared to the case where the entire sensor set on the grid is sampled at all times.

Categories and Subject Descriptors: C.5.3 [**Computer System Organization**]: Computer Systems Implementation—*Microcomputers and microprocessors—thermal management*

General Terms: Design, Measurement, Performance

Additional Key Words and Phrases: Thermal sensor allocation, nonuniform and uniform sensor placement

This work has been supported by SRC Grant No. 1660.001 and NSF Grants CNS-0546305, CCF-0747201, and CCF-0541337.

Authors' addresses: Jieyi Long, Seda Ogrenci Memik, and Gokhan Memik, Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois; 60201; email: jieyi-long@northwestern.edu or jlo198@eece.northwestern.edu; Rajarshi Mukherjee, Synopsys, Mountain View, California 94043.

ACM Reference Format:
Long, J., Memik, S. O., Memik, G., and Mukherjee, R. 2008. Thermal monitoring mechanisms for chip multiprocessors. ACM. Trans. Architec. Code Optim. 5, 2, Article 9 (August 2008), 33 pages. DOI = 10.1145/1400112.1400114 http://doi.acm.org/10.1145/1400112.1400114

1. INTRODUCTION

Steady miniaturization and large-scale integration are vital to meet aggressive performance targets in high-performance microprocessors. However, these advances have led to rapidly increasing power densities on microprocessors [Borkar 1999]. Power dissipated on a chip is converted to heat. Heat, in turn, creates reliability threats, adversely impacts leakage, and increases cooling cost. One of the challenges in semiconductor industry in the nanoscale era is to provide high performance and reliability at lowest cost possible.

Effective assessment and analysis of the thermal behavior is crucial to overcome this challenge. Modeling and simulation tools and thermal-aware design methodologies are one avenue of efforts toward this goal [Cong et al. 2004; Huang et al. 2004; Sankaranarayanan et al. 2005; Skadron et al. 2003]. A major hurdle in this direction is the fact that thermal behavior is input dependent and sensitive to environmental conditions. Thus, a highly accurate thermal profile of a complex system can only be established after it is deployed.

Thermal monitoring using on-die thermal sensors provides the means to assess the runtime thermal profile of a system. Thermal monitoring is already in use in modern processors 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 [Rotem et al. 2004].

Accuracy is crucial for thermal monitoring. Overestimation of temperature impacts performance negatively because of unnecessary triggering of performance throttling, e.g., dynamic voltage and frequency scaling (DVFS). It has been reported that in mobile computers, 1.5°C accuracy in temperature measurement is equivalent to 1 W of CPU power and in desk-top computers 1°C accuracy translates into 2 W of CPU power [Rotem et al. 2006]. On the other hand, it has been shown that the mean time to failure (MTTF) decreases exponentially with increase in temperature [Srinivasan et al. 2004]. Therefore, underestimation of the die temperature is harmful since the processor will continue to operate at a higher temperature than its rated operating condition.

Current sensor implementations are capable of providing accuracy levels around 1°C with a corresponding trade-off in size and complexity of the sensor design. Furthermore, increasing the number of sensors deployed can maximize accuracy. Although there is a clear trend in elevating the number of sensors in commercial microprocessors, increasing the number of sensors arbitrarily will come with several overheads and the construction of sensor networks are subject to constraints from various sources.

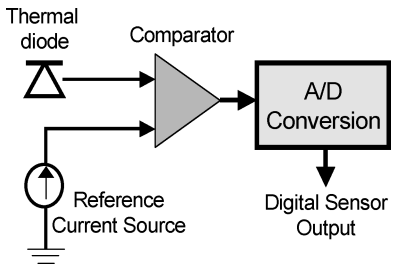

Fig. 1. Digital thermal sensor.

1.1 Hardware Cost

There are two types of temperature sensors: analog and digital. Analog sensors can produce a voltage or current proportional to temperature, exploiting the fact that many electrical parameters of semiconductor devices are sensitive to temperature variation. Examples of these parameters include PN-junction forward voltage, threshold voltage, leakage current, and gain [Blackburn 2004]. An analog thermal sensor can be built using a temperature-sensing diode, a factory-calibrated reference current source, and a current comparator (see Figure 1).

For highest accuracy it would be ideal to place the thermal diode as close to hotspots as possible. However, other components, such as the reference current source, are sensitive to temperature variation. One solution is to place the thermal diodes as near the hotspots as possible while the core of the thermal sensor resides near an area that is better suited for the sensitivities of the analog components. This is referred to as *remote sensing*. The total area requirement for a basic analog sensor is not significant. However, remote sensing leads to other overheads. First, the outputs of all thermal diodes need to be routed to another location for postprocessing and analog-to-digital conversion. After all, the control mechanism for DTM requires a digital input. Dorsey et al. [2007] discussed the thermal monitoring mechanism of AMD quad-core Opteron processor. As shown in Figure 2, the thermal sensors are scattered across four cores, while the sensor data processing is centralized per core. These processing centers are called thermal evaluation circuits (TCEN). It can be easily observed that sensor data need to be transmitted across large distances. Several sensors require routing of data across the entire core. Similarly sensor data is routed across the length of the interconnect network.

Other types of sensors are more sophisticated [Quenot et al. 1991; Wang et al. 2003; Tuthill 1998]. They usually include a serial interface, such as I^2C, SPI, or SMBus, which provides communication with embedded microcontrollers and other digital systems [http:// www.maxim-ic.com]. Additional circuitry is needed to digitize the analog signals. Many new generation processors employ multiple on-chip digital thermal sensors of this type (such as IBM Power5). The accuracy and stability of digital sensors can be enhanced by increasing their sizes, which will emphasize the constraints on hardware resources.

There are other emerging types of digital sensors. A 4T-decay sensor has been proposed, which consists of a 4T-memory cell and a decay counter [Kaxiras and

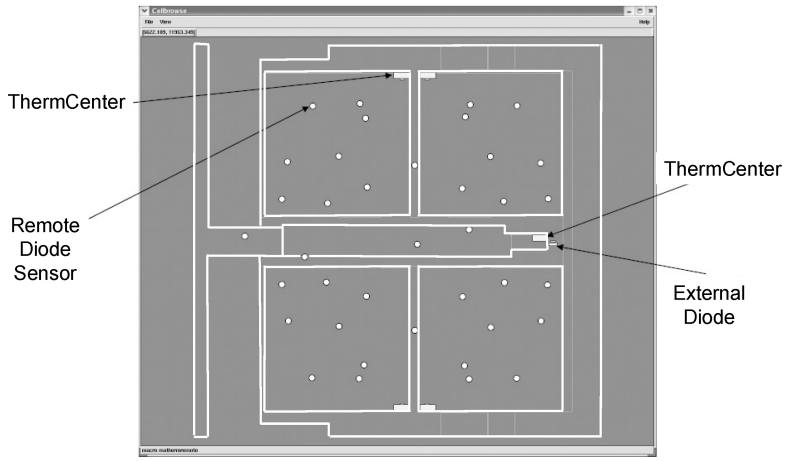

Fig. 2. AMD quadcore Opteron TM processor floorplan. Locations of thermal diodes and thermal processing centers are depicted [Dorsey et al. 2007].

Xekalakis 2004]. The 4T-cell senses the temperature and converts it to a digital signal. Its area and power consumption are 0.0017 mm^2 and 221 μW, respectively (as reported at 180-nm technology [Kaxiras and Xekalakis 2004]). The decay counter performs the average operation on the digital signal to improve the accuracy. The larger the counting range, the higher the sensor precision. For example, to achieve $+/-1^{\circ}$C of accuracy, a 7-bit decay counter is needed, which occupies about 0.0016 mm^2 area (doubling the initial area) and consumes 397 μW (tripling the initial power consumption) power (at 180-nm technology, according to our evaluation using the synopsys design compiler synthesis tool).

1.2 Power Consumption

In the AMD quadcore opteron case, the sensors are diodes and the sensor data are analog signals (continuous current). Let us assume the current of a diode is 1 mA [http:// www.capgo.com], the resistivity of the interconnect is 1.68 e–8 ohm·m (copper), and the width and thickness of the interconnect are 270 and 135 nm (assuming thickness/width ratio is 0.5), respectively. Based on the die photo [Dorsey et al. 2007] and the sensor placement reported, on average, the Manhattan distance between a sensor and its associated TCEN can be estimated as 8 mm. As there are 38 sensors in total, the length of the interconnect dedicated for sensor data transmission is as long as 304 mm. Based on the above parameters, the power consumption for transmitting the analog sensor reading is 140 mW. If we add analog signal amplifiers along this path, the total power will increase further. As the number of cores per chip and the number of sensors per core increases (as predicted by the current trends), the power consumed for thermal sensor data can easily reach tens of watts, which is significant. In addition, note that other support circuitry, depending on the sensor type (A/D conversion, comparator, counters, and registers), will also consume additional power.

1.3 Stability Issues and Calibration

One disadvantage of thermal diodes 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. Process variations can have a large impact on these parameters. In fact, each processor and each sensor within a processor needs to be individually calibrated during manufacturing to eliminate any potential manufacturing variations [http:// www.download.intel.com]. Higher number of on-chip cores and more sensors per core will directly impact the time consumed by this calibration stage.

One of the parameters used to determine the core temperature is the *diode ideality factor*, which describes the behavior of the diode relative to a theoretically perfect diode. The ideality factor depends on the characteristics of each individual processor and will vary slightly from one chip to the next. The interpretation of sensor data should take this range into account in order to improve the accuracy of the reading. Another variable that influences accuracy is the *series resistance*, which is a measure of the resistance in the paths leading up to and away from the thermal diode. Some diode sensors have the ability to adjust for the temperature error (which, in turn, incurs additional area overhead), while some do not. Furthermore, routing sensor data cross-core to central processing units will exacerbate this effect. Finally, note that, for remote sensing, analog signals will travel large distances across cores. The noise induced on the thermal diode paths by high-speed signals in the surrounding is yet another source of inaccuracy.

In summary, if remote sensing is used, routing sensor data to far central units will incur significant overheads in terms of area, power, and sensor stability. Embedding a large number of digital sensors into the hotspot locations will be challenging, since the layouts of components are highly optimized. Clearly, the number and location of sensors used by the monitoring infrastructure will have a direct impact on the implementation overheads and power cost, as well as on the calibration effort. This motivates the need to carefully optimize the allocation and management of thermal sensors.

In this paper, we target this increasingly important problem. Our goal is to provide accurate temperature readings in a given system while maintaining a reasonable overhead in terms of number of sensors. In this context, we investigate a wide range of sensor-allocation techniques and their respective effectiveness. Particularly, we first extend thermal-aware K-means approach [Mukharjee and Memik 2006] for CMPs. Observing the limitations of this approach in CMPs, we analyze grid-based methods to distribute sensors on a chip and analyze their performance while varying the number of sensors allocated for each core. Third, we introduce an interpolation scheme that estimates the temperature based on the sensor readings. Finally, we devise a dynamic sensor-selection mechanism, which activates only a small fraction of the sensors on a chip and processes their readings at a given time. This filtering mechanism can correctly select the proper subset of sensors that give the best representation of the chip temperature.

The thermal behavior of complex systems is affected by various factors. Power dissipated by different resources can be spatially nonuniform across the entire

system. Power-management techniques, such as local clock gating, further create disparity in power densities among different regions on a chip. These phenomena lead to significant on-chip temperature gradients. For instance, based on the thermal map provided by Intel researchers [Borkar et al. 2003], on-chip temperature difference can be as large as 50°C. Other industrial chips exhibit even steeper thermal gradients [Tsai et al. 2006]. Besides, chip temperature profiles may also vary dramatically over time. With the increasing impact of process technologies and associated process variations, it becomes increasingly harder to statically predict the hotspots of a processor. For example, because of increased leakage consumption of a functional unit, it can become a hotspot instead of another component. It is reported that the single thermal sensor on the Intel Pentium 4 processor is placed near the rapid-integer ALU, which was identified as the most likely candidate to cause a hotspot [Krinitsin]. Skadron et al. [2003] report that in the Alpha 21364 architecture the register file appears to be the hottest component consistently across a large set of SPEC CPU2000 benchmarks. In addition to these varying results, the high number of sensors utilized in other architectures (e.g., IBM Power5), is an indication of the fact that it is very challenging to have precise temperature readings in modern processors by static estimations of hotspots. More importantly, the shift toward CMP paradigm complicates thermal monitoring even further. In such architectures, the thermal profiles can become even less predictable because of the thermal coupling between cores. In fact, we analyze a thermal sensor placement mechanism developed for single-core processors and show that it provides limited accuracy for CMPs. To make matters worse, a study of thermal behaviors of different SMT and CMP architectures reveals that CMPs will generally be hotter than SMT machines in future process technologies, which motivates accurate temperature monitoring in CMPs [Li et al. 2005].

In conclusion, the thermal behavior of a microprocessor can be affected by a variety of factors. Hence, there is a clear need to establish a thermal monitoring mechanism that can capture the thermal behavior with high fidelity. In this paper, we address this challenge. Our specific contributions in this paper are as follows:

- Analyze clustering-based and grid-based algorithms for CMP sensor allocation and placement,
- Propose an interpolation scheme to predict the maximum temperature,
- Propose a dynamic sensor-selection scheme to avoid collecting data from those sensors that will not provide useful information, and
- Present simulation results, studying the impact of different placement strategies on measurement accuracy.

In the following, we give an overview of related work. In Section 3, we introduce the multicore architecture and elaborate on its thermal characteristics. We discuss the nonuniform sensor placement by thermal aware K-Means clustering in Section 4. The interpolation and dynamic sensor selection techniques are presented in Section 5. Section 6 presents our experimental results we conclude with a summary in Section 7.

2. RELATED WORK

The sensor allocation on microprocessors is a largely unexplored problem. Lee et al. [2005] presented an analytical model that describes the maximum temperature difference between a hotspot and a region of interest. Gunther et al. present observations on thermal maps of a microprocessor and point to opportunities for optimized decisions on sensor placement, however, they do not provide any solutions [Gunther et al. 2001].

On a different track, Bratek and Kos [2001] present sensor placement for fault diagnosis of integrated circuits, by linking temperature sensors and power modules in pairs. Lopez-Buedo et al. [2002] investigated instantiating digital sensors on field-programmable gate arrays (FPGAs). Velusamy et al. [2005] used such digital sensors to validate the accuracy of the thermal simulator HotSpot in modeling a FPGA-based system. Mondal et al. [2006] reported a sensor-insertion scheme for FPGAs. In their approach, the hotspot-monitoring problem was reduced to a set-covering problem for a well characterized set of hotspot locations. Mukherjee et al. [2006] proposed another algorithm for reconfigurable systems. This divide-and-conquer-based algorithm locates vacant configurable logic blocks that can be used to instantiate thermal sensors and embeds them into a design mapped onto a FPGA. Obviously, the architectural constraints for reconfigurable systems are fundamentally different than CMPs. Mukherjee and Memik [2006] also presented a thermal sensor allocation scheme for microprocessors. However, their approach only addresses single-core processors. Moreover, they propose a profile-driven placement of sensors into nonuniform locations and do not offer any placement methods where sensor locations are not dependent on a thermal profile. Such schemes can be inaccurate because of the interaction of cores, which cannot be accurately predicted statically. In fact, we modify this approach for CMPs and show its limitations. Furthermore, we contribute new placement strategies that do not depend on any a priori knowledge on workload or thermal profiles.

3. UNDERLYING ARCHITECTURE AND ITS THERMAL IMPLICATIONS

In this paper, we analyze two different CMP architectures, which are shown in Figure 3. The first architecture consists of 16 cores, distributed in a 4×4 array. Each core includes local level-1 data and instruction caches. The level-2 cache, on the other hand, surrounds the array of cores and it is shared by all the cores. This architecture will be referred to as the *dense architecture* throughout the rest of the paper. The floorplan of the second architecture is different from the first one in that part of the level-2 cache is dispersed between the cores. This architecture will be called the *sparse architecture* in the remainder of the paper. Throughout the execution, the distribution of threads to different cores can lead to uneven amounts of activity in different cores. The slow lateral heat propagation in silicon creates localized heating zones, hence, hotspots in certain cores. We must note that the task distribution on such systems will be performed primarily for performance and communication constraints, since interconnect delay will be a major component of performance. Therefore, for these systems, task distribution is unlikely to present a thermally

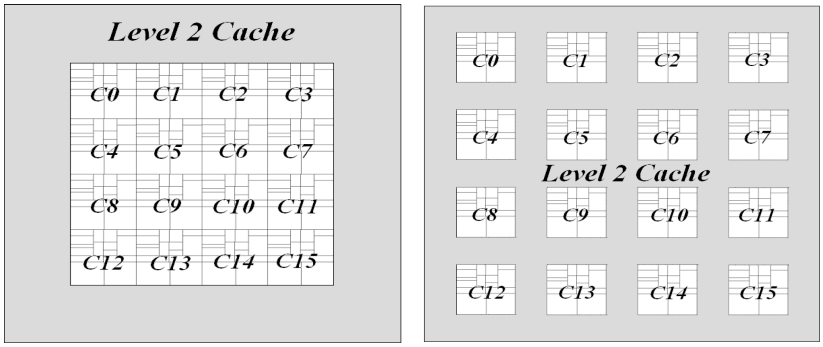

Fig. 3. Floorplans of the CMP chip for (a) the dense architecture with L2 cache surrounding the array of cores and (b) the sparse architecture with L2 cache distributed between the cores.

good solution and localized activity will prevail, leading to thermal emergencies. Nevertheless, thermal-aware task distribution will certainly be an important component of the thermal-aware system design paradigm and our temperature measurement techniques would coexist with them in a comprehensive solution.

Instantiating multiple cores on a single chip improves performance per watt efficiency [Phan et al. 2005] and it is predicted that we are headed into multicore processor era [Borkar et al. 2005]. As new architectures with increasing number of cores are developed, the arrangement of the cores in the layout is changing. For instance, Kongetira et al. [2005] presented Niagara, a 32-way multithreaded SPARC processor. The processing pipeline consists of eight Sparc pipes, which are stacked immediately next to each other. The first generation cell processor has a 64b Power core and eight synergistic processor elements residing next to each other [Phen et al. 2005]. Intel IXP2800 has sixteen independent microengines arranged in an array and one XScale core [Borkar et al. 2005]. Further, Intel is experimenting with tens of cores, potentially even hundreds of cores per single die [Rattner 2005]. These trends indicate that in future architectures there is a greater likelihood of the cores being placed physically adjacent without L2 caches thermally insulating them. Our first architecture, depicted in Figure 3(a), provides a model for these kinds of CMPs. In such architectures, the thermal behaviors of cores will not only depend on the internal activity within the core, but it will also be shaped by the physical interaction between individual cores. For example, consider the interaction between cores C6 and C10 in Figure 3(a). At a given time during the execution, the integer execution unit of core C10 becomes a hotspot. Because of lateral heat propagation it can cause the hottest point of C6 to lie in its data cache (placed immediately adjacent to the border from C10) even if the application mapped on C6 has more accesses to other units, say, the floating-point units. On the other hand, architectures where parts of the L2 cache are placed between the cores are also plausible. Powell et al. [2004] assumed that the SMT cores in a CMP are thermally insulated by L2 cache blocks. The thermal effects of the adjacent blocks were not considered by Huang et al. [2000], either.

Table I. Operating Frequency and Voltage Levels

Freq [GHz]	2.5	2.13	1.86	1.6	1.46	1.33	1.2	1.06	0.8
V_{dd} [V]	1.476	1.372	1.292	1.212	1.18	1.148	1.1	1.068	0.988

Kumar et al. [2005] presented floorplans for 4-, 8-, and 16-core CMPs. The thermal behavior of such chips can be captured using our second architecture.

The dynamic power consumption of a core due to switching activity is given by

$$P_{\text{dynamic}} = 0.5\alpha C\, V_{dd}^2 f$$

where α is the switching activity of the core, and V_{dd} and f are the operating voltage and frequency, respectively. Power consumed is dissipated as heat, which leads to a rise in temperature. As temperature rises, DTM methods are activated to prevent the temperature from reaching critical ranges. Although DTM is not directly related to our schemes, our evaluation environment implements a DTM scheme that utilizes dynamic voltage and frequency scaling (DVFS). The reason for implementing DTM lies in our goal of observing the sensor error readings in realistic temperature ranges. Without DTM, a core can become excessively hot, making comparisons inaccurate and unrealistic. Our operating frequency and voltage levels for the cores are assumed to be similar to the Intel Pentium M Sonoma specifications. We also assumed that DVFS can be independently applied to each core. Similar assumptions about independent frequency and voltage control of each core can be found in literature [Juang et al. 2005]. The different frequency and corresponding voltage levels are shown in Table I. We set the *emergency temperature level* to 82°C. When the maximum temperature of a core reaches 82°C, we start reducing the frequency of the core. For each degree above 82°C, we reduce the operating frequency by one level. For example, if the temperature is 82°C, we reduce the frequency to 2.13 GHz. If the temperature is 83°C, the frequency is reduced to 1.86 GHz, and so on. Once the core temperature reduces below 82°C, it then runs at full speed (2.5 GHz). Note that, as shown in Table I, along with the frequencies, we also change the supply voltage of the core, which significantly reduces the power consumption.

Leakage power is also accounted for in our power model. We calculate the leakage power based on the dynamic power and chip thermal profile via an iterative method [Liu et al. 2007].

Our investigation for effective sensor placement has been conducted under the aforementioned assumptions on the physical layout and operating conditions. We also assumed a microcontroller-based collection and processing of sensor data similar to the Foxton [Poirier et al. 2005]. The sensor-placement methods we have examined can be categorized into nonuniform and uniform. Nonuniform sensor locations can be determined through optimization of an objective function, representing the relation between the locations of thermal events and the proximity of sensors to those locations. Uniform sensor locations are formed statically by dividing a core using a uniform grid and placing sensors on the grid points. The nonuniform scheme uses profiling on thermal behavior of applications. Sections 4 and 5 describe the methodologies we have developed for each case.

4. NONUNIFORM SENSOR PLACEMENT

Mukherjee and Memik [2006] proposed a nonuniform sensor placement method called thermal-aware K-means algorithm for single-core processors. We have adapted this algorithm to CMPs. The goal is to systematically analyze thermal maps generated from profiling data to identify locations, which lie close to a maximal number of eventful thermal spots across a range of applications. The problem is formulated as a clustering of the points of interest in the spatial domain. The center of each cluster will indicate the physical location of a sensor. The reading from that sensor is representative of its respective coverage area.

We define a hotspot as a point, which reaches the highest temperature during the execution of an application. This point can be defined *globally* over the entire processor (to monitor reliability threats) or *locally* for each core (to assist in dynamic thermal management of individual cores). In our study, the hotspots are assumed to be defined locally. If we consider a set of applications mapped onto our multicore architecture, each application can create a distinct hotspot in its execution core with a distinct temperature value. The location and temperature of the hotspot can also change dynamically during the execution of a single application. Considering all of these possible events yields a set of hotspots observed across a range of applications as well as over time. Such a set of hotspots can be obtained via thermal simulation of a given architecture configuration. We will describe our methodology to perform such a simulation in Section 6. The thermal-aware K-means clustering is then applied onto this map to identify a minimal set of sensors that can monitor the thermal behavior of the given map with a defined error margin.

We collect the hotspots whose temperature is above the emergency temperature level (82°C). Figure 4 shows the distribution of these hotspots in the dense architecture when different sets of applications are executed over a period of time. The hotspots move into different components of the cores. For example, the floating-point register can be the hottest point in some applications, whereas for another application the hotspot can move into the integer-execution unit. Hotspots can also move into different locations during the execution of one application. The hotspots for each core over the execution periods of all applications form the input for clustering. We first fold the hotspots of all the cores into a single core. This means that all types of hotspot formations that can be observed across 16 cores are superimposed onto one thermal map. The sensor placement decision will then be made on this thermal map. As a result, any given core will be equipped with the best possible sensor allocation ready for any of the encountered hotspot distributions. The folded hotspot maps for both architectures are shown in Figure 5. In this example, for the dense architecture, the input to the K-means clustering contains 667 hotpots. The temperatures of these hotspots range between 117° and 82°C. The sensor locations for placing 16 sensors obtained by applying clustering method on both architectures are shown later in Figure 11.

In our experiments, multiple sensors are placed per core to accurately monitor the hotspots. We perform such placement by varying the number of clusters in thermal-aware K-means clustering. A major limitation of K-means-based

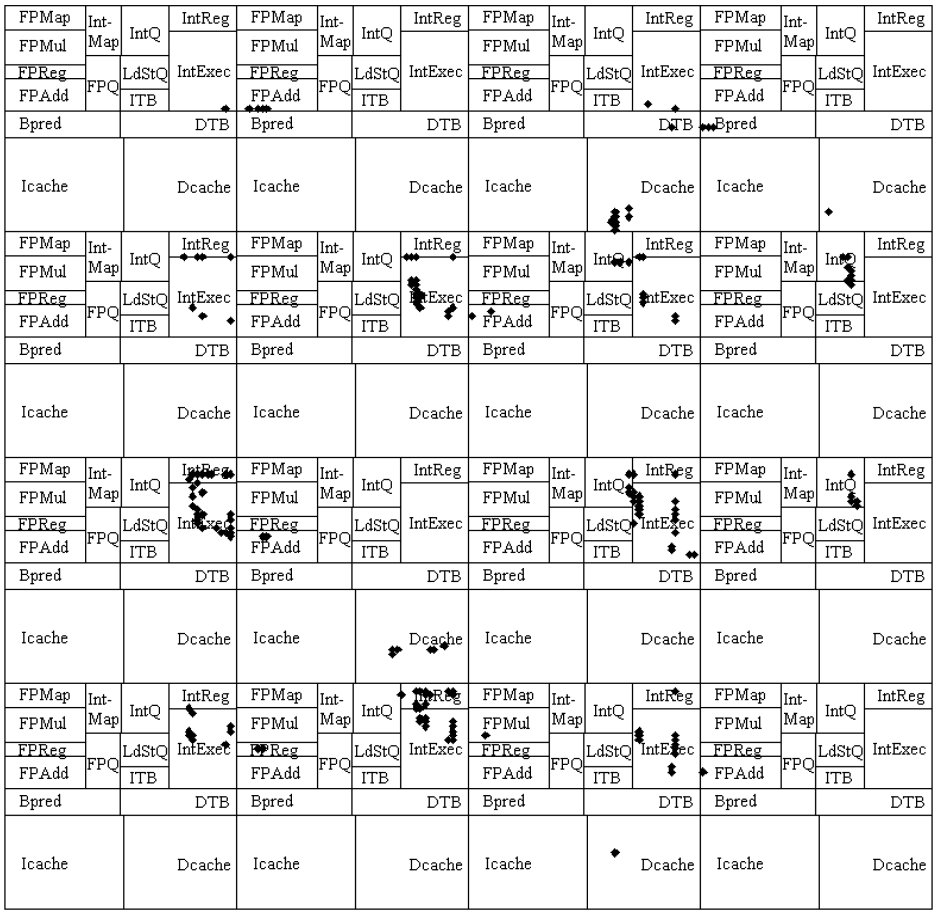

Fig. 4. Distribution of hotspots for all application sets in the dense architecture.

algorithm is its dependence on profiling data. In addition, as will be shown in Section 6, the thermal-aware K-means algorithm does not work well for the dense architecture, where the cores have strong thermal impact on each other. To address these problems, we have developed several uniform sensor placement methodologies, which are explained in the next section.

5. UNIFORM SENSOR PLACEMENT

Since it is hard to statically predict the exact locations of hotspots on a chip, a possible methodology is to divide each core into equal-sized quadrants and place a sensor at the center of each quadrant (these sensors will be referred to as *grid sensors*). All grid sensors then work in parallel and the maximum temperature measured among them will be used as the estimation of the core temperature. Although this is fairly straightforward to apply, as we will show in Section 6.2, it exhibits large inaccuracies. Particularly, for the dense architecture, sensor readings can differ from the actual maximum core temperature

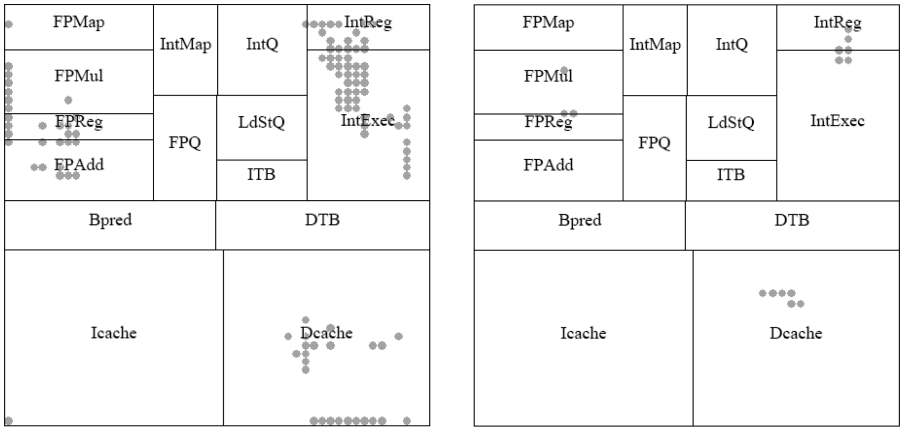

Fig. 5. Hotspots across different benchmarks and iterations folded onto a single core for (a) the dense and (b) the sparse architecture.

by as much as 12.6°C when using 16 sensors per core. Also, as large as 10.6% of the thermal emergencies are not captured using the same number of sensors. Even with 36 grid sensors per core, the maximum error can still be as large as 9.0°C. Unfortunately, placing an arbitrary number of sensors on the chip is not the best solution. Therefore, it is beneficial to maintain an upper bound on the number of sensors. At the same time, the grid-based approach described above fails to perform reliably even if we dedicate a very high number of sensors. These observations motivate us to investigate more effective techniques for sensor allocation. First, we present 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. Second, we present a methodology to dynamically select a subset of sensors to provide readings from a large number of physical sensors.

5.1 Grid-Based Interpolation Method

The relatively large errors in the basic uniform grid are because of the fact that we cannot control the distances of the sensors to the hotspots by a static placement. A corrective measure is needed to further refine the readings obtained from the uniform placement. We have developed an interpolation scheme to estimate hotspot temperatures using grid sensor readings. For each sensor, we use its readings and those of its neighbors to estimate the position and temperature of the hottest spot within its neighborhood. The neighborhood of a sensor S_i is defined as $N(S_i) = \{(x, y)|x_i - r_s/2 \leq x \leq x_i + r_s/2, y_i - r_s/2 \leq y \leq y_i + r_s/2\}$, where (x_i, y_i) is the x-y coordinates of S_i, and r_s is the distance between two consecutive sensors. This neighborhood represents a square of width r_s with sensor S_i at its center. Considering the sensors shown in Figure 6, the neighborhood of S_4 is the region inside the dashed square, whose sides extend midway between S_4 and its immediate neighbors, S_1, S_3, S_5, and S_7. Once we have the maximum

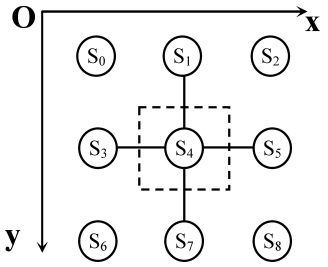

Fig. 6. Illustration of the interpolation scheme.

temperature estimation of the neighborhood of each sensor on a core, we will use the maximum among them as the core's peak temperature estimation.

Consider x direction first. In Figure 6, if the reading of S_5 is larger than those of S_3 and S_4, the hottest spot within $N(S_4)$ should be close to the midpoint between S_4 and S_5. Particularly, if the gradient of the thermal curve does not change between S_4 and S_5, the x coordinate of that hottest spot should be $x_4 + r_s/2$. Likewise, if the reading of S_3 is larger than those of S_4 and S_5, the x coordinate of the hottest spot within $N(S_4)$ can be estimated as $x_4 - r_s/2$. If the reading of S_4 is larger than those of S_3 and S_5, we use both gradients between S_3 and S_4 and, S_4 and S_5 to perform the estimation. We assume that the temperature gradient is constant within the segment between the hottest spot and the nearest sensor, i.e., in that segment the temperature can be approximated as a linear function of x and y coordinates. Assuming the reading of S_3, S_4 and S_5 are T_3, T_4, and T_5, the x coordinate of the hottest spot within $N(S_4)$ should be

$$x_4 + \frac{1}{2}\frac{\Delta T_3 - \Delta T_5}{\Delta T_3 + \Delta T_5} r_s$$

where $\Delta T_3 = T_4 - T_3$, $\Delta T_5 = T_4 - T_5$. Similar arguments apply to the y direction. In general, the coordinates of the hottest spot within $N(S_4)$ can be estimated using Equations (1) and (2). We then, substitute the estimated coordinates of the hottest spot into the linear function representing temperature within the limited segment between the hottest spot and the nearest sensor location. This substitution results in Equations (3) and (4), which are used to obtain the value of the estimated maximum temperature.

$$\Delta x = \begin{cases} \frac{1}{2}\frac{\Delta T_3 - \Delta T_5}{\Delta T_3 + \Delta T_5} r_s \ldots\ldots\ldots T_4 \geq \max(T_5, T_3) \\ -\frac{1}{2} r_s \ldots\ldots\ldots\ldots\ldots\ldots T_3 \geq \max(T_4, T_5) \\ +\frac{1}{2} r_s \ldots\ldots\ldots\ldots\ldots\ldots T_5 \geq \max(T_3, T_4) \end{cases} \quad (1)$$

$$\Delta y = \begin{cases} \frac{1}{2}\frac{\Delta T_1 - \Delta T_7}{\Delta T_1 + \Delta T_7} r_s \ldots\ldots\ldots T_4 \geq \max(T_7, T_1) \\ -\frac{1}{2} r_s \ldots\ldots\ldots\ldots\ldots\ldots T_1 \geq \max(T_4, T_7) \\ +\frac{1}{2} r_s \ldots\ldots\ldots\ldots\ldots\ldots T_7 \geq \max(T_1, T_4) \end{cases}$$

$$(x_{est}, y_{est}) = (x_4 + \Delta x, y_4 + \Delta y) \tag{2}$$

$$\Delta T_x = \begin{cases} \dfrac{|\Delta T_5 - \Delta T_3|}{\Delta T_5 + \Delta T_3} \max(\Delta T_5, \Delta T_3) \text{.....} T_4 > \max(T_5, T_3) \\ \dfrac{\max(T_5, T_3)}{2} - T_4 \text{.....} T_4 \le \max(T_5, T_3) \end{cases}$$
$$\Delta T_y = \begin{cases} \dfrac{|\Delta T_7 - \Delta T_1|}{\Delta T_7 + \Delta T_1} \max(\Delta T_7, \Delta T_1) \text{.....} T_4 > \max(T_7, T_1) \\ \dfrac{\max(T_7, T_1)}{2} - T_4 \text{.....} T_4 \le \max(T_7, T_1) \end{cases} \tag{3}$$

$$T^{\max}_{N(S_4)} = T_4 + \Delta T_x + \Delta T_y \tag{4}$$

Equations (1) through (4) are based on the assumption that the gradient of the temperature curve remains constant between S_4 and the estimated hottest spot. However, on a real chip, this may not be the case. Figure 7 portraits a temperature surface of the dense architecture. It clearly shows that the temperature gradient can change significantly. Especially within the regions surrounding the maximal temperature points, where we are primarily interested in, the thermal gradient can change from some large value to zero within a small distance. Therefore, the peak temperature could be largely overestimated if we apply Equations (3) and (4) directly. Therefore, we modify Equation set (3) to obtain the Equation set (5).

$$\Delta T_x = \begin{cases} \kappa \dfrac{|\Delta T_5 - \Delta T_3|}{\Delta T_5 + \Delta T_3} \max(\Delta T_5, \Delta T_3) \text{.....} T_4 > \max(T_5, T_3) \\ \beta\left(\dfrac{\max(T_5, T_3)}{2} - T_4\right) \text{.....} T_4 \le \max(T_5, T_3) \end{cases}$$
$$\Delta T_y = \begin{cases} \kappa \dfrac{|\Delta T_7 - \Delta T_1|}{\Delta T_7 + \Delta T_1} \max(\Delta T_7, \Delta T_1) \text{.....} T_4 > \max(T_7, T_1) \\ \beta\left(\dfrac{\max(T_7, T_1)}{2} - T_4\right) \text{.....} T_4 \le \max(T_7, T_1) \end{cases} \tag{5}$$

Equation set (5), along with Equation (4), is used for the temperature estimation. The physical meaning of κ is illustrated in Figure 8(a). When $T_5 > T_3$, κ is the average thermal gradient between x_4 and x_{est} divided by the average gradient between x_3 and x_4 (if $T_3 > T_5$, κ should be the average gradient of the thermal curve between x_4 and x_{est} divided by the average gradient between x_5 and x_4). The physical explanation of β is similar.

We assume the chip is thermally isotropic, so we use the same κ and β for both x and y directions. Generally speaking, calculating the value of κ and β is nontrivial, since it is hard to find an exact equation to fit the temperature–distance curve [Lee et al. 2005]. However, it can be seen from Figure 8 that κ should be close to zero since the gradient of the temperature curve quickly approaches to zero near the maximal point. On the other hand, β should be close to one, as the gradient does not change much between x_4 and x_5. We performed several tests with varying values of κ and β in our experiments. Although κ

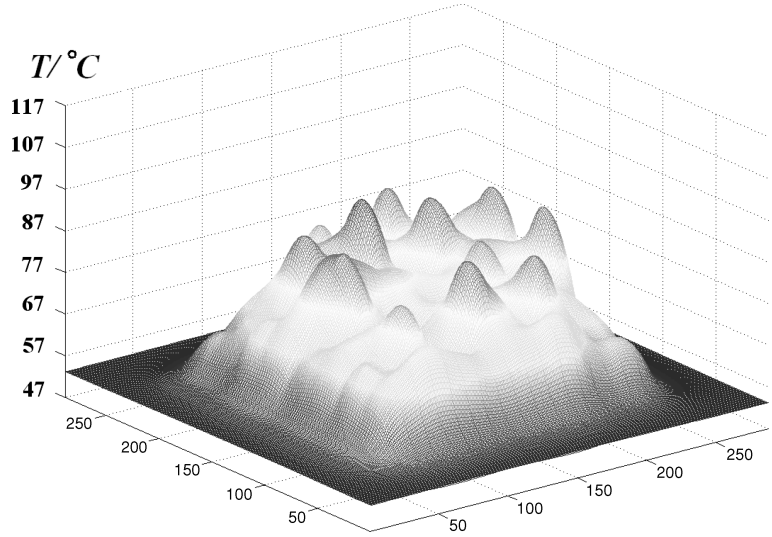

Fig. 7. Dense architecture thermal profile example.

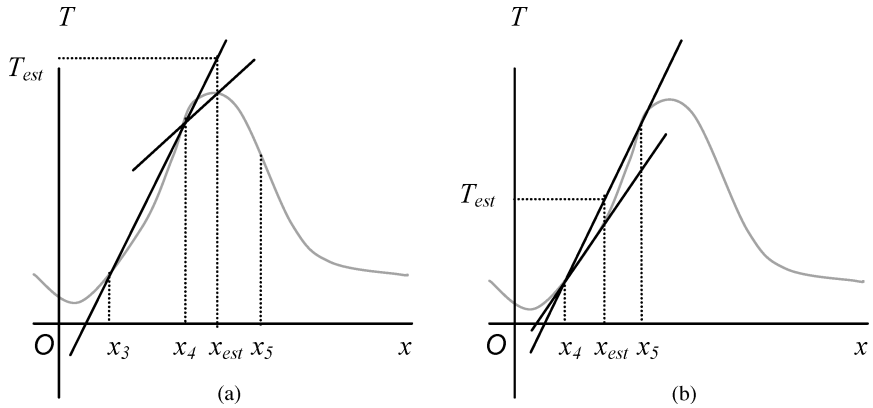

Fig. 8. Physical depiction of (a) κ and (b) β.

and β are complicated functions of sensor location (x_i, y_i) and sensor distance r_s, and are architecture dependent, we found that setting κ to $0.83/\sqrt{N_{sensor}}$ and β to 0.93 has given us the smallest estimation error (N_{sensor} is the number of sensors allocated per core).

5.2 Interpolation-Based Dynamic Selection Scheme

In many processor families (e.g., PowerPC, Intel Core Duo, Cell Processor), digital sensors are being employed. Thermal sensor readings are routed to a central microcontroller for further thermal management. Processing data collected by a large number of digital sensors presents challenges. Instantaneous communication across multiple global communication lines will likely incur significant interconnect power overhead. Reducing the number of sensors involved in data communication may help relieve this overhead. However, this will cause

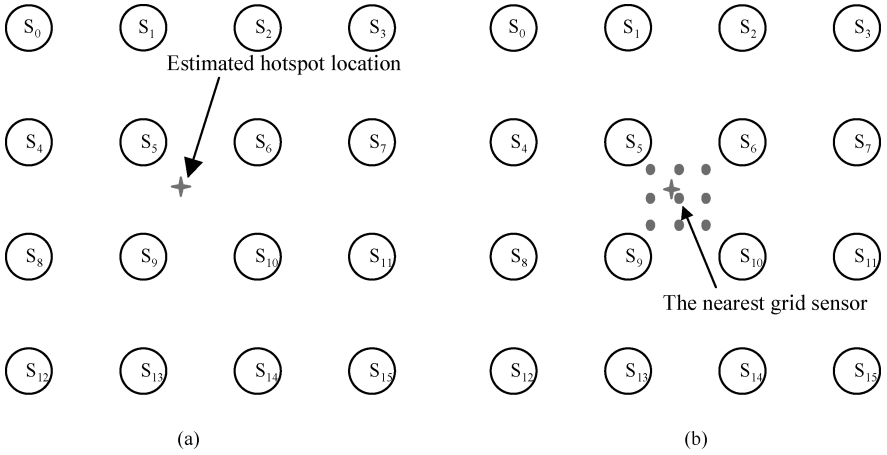

Fig. 9. Steps for the interpolation-based dynamic selection method. (a) 16 coarse-grain sensors activated. Hotspot location estimated based on the coarse-grain sensor reading. (b) The grid sensor closest to the estimated hotspot and eight surrounding sensors are activated.

inaccuracies; as we will further illustrate in Section 6.2, a slight increase in sensor distance r_s (equivalent to reducing the number of sensors) could cause large measurement errors. In order to address the problem of achieving optimal accuracy with minimum sensor data communication and processing, we have investigated an alternative approach. We have developed an interpolation-based dynamic selection scheme that embeds a large number of grid sensors into each core, but receives readings from only a small subset of them at a given instant. The idea is to create a hierarchy of sensors and first activate coarse-grain sensors and use their readings to estimate the hotspot location of the core using the technique illustrated in Section 5.1. The grid sensor closest to the estimated hotspot location is then activated. To further increase the accuracy, eight sensors surrounding this sensor are also activated since the location estimation may not be precise. The maximum among the readings of these nine sensors are used as the estimated hotspot temperature of the core. By directly reading the grid sensor instead of calculating temperature using Equations (4) and (5), the estimation error caused by the inaccuracy of κ and β is eliminated. As a result, this method adapts better to different architectures.

The methodology is depicted in Figure 9. The total number of the grid sensors is 64 (only a subset is shown here). We first divide the chip into 16 equal sized quadrants and use grid sensors S_0 through S_{15}, which reside at the center of the quadrants, as the coarse-grain sensors. These coarse-grain sensors are activated in the first stage. Their readings will be fed into Equations (1) and (2) to estimate the hotspot location. In Figure 9(a), the estimated hotspot location is marked with a star. In the second stage, the grid sensor that is closest to the estimated location and the eight sensors surrounding it are activated, as shown in Figure 9(b). The maximum readings from nine activated sensors will be used as the hotspot temperature estimation. In this example, the total number of activated sensors is only 25 (16 coarse-grain sensors plus 9 grid sensors that are activated later).

Table II. Categories of Benchmarks and Their Average IPC Values

High IPC Benchmarks		Medium IPC Benchmarks		Low IPC Benchmarks	
Benchmark	*IPC*	*Benchmark*	*IPC*	*Benchmark*	*IPC*
mesa	1.8092	bzip2	1.2689	ammp	0.55
galgel	1.8065	gap	1.2352	applu	0.544
gzip	1.7667	sixtrack	1.0216	equake	0.3594
perlbmk	1.5908	facerec	0.7616	art-470	0.2135
apsi	1.5095	mgrid	0.6885	mcf	0.0292

6. EXPERIMENTAL RESULTS

In the following, we first describe our experimental methodology. In Section 6.2, we present our results for different sensor allocation strategies. Section 6.3 summarizes the results and compares the techniques.

6.1 Experimental Setup

We simulated the SPEC2000 benchmark suite [SPEC-CPU 2000] using M5 Simulator [Binkert et al. 2006]. M5 is a general-purpose architecture simulator employing a detailed performance model of the CPU, memory, and I/O subsystems [Binkert et al. 2006]. We have considered two different architectures, as shown in Figure 3(a) and (b). The first one has 16 cores arranged in a 4×4 array with level-2 cache surrounding them. In the second architecture, the level-2 cache is placed in between the cores. In both architectures, each core is modeled as an Alpha 21364 processor, which is a four-way processor with a load store queue and register update unit of sizes 64 and 128, respectively. The level-1 instruction and data caches are 64 KB, four-way associative with 32-byte block size and two cycle latencies. Unified level-2 cache is 8 MB, eight-way associative with 128-byte line size and has a latency of 13 cycles.

First, we determine the instructions per cycle (IPC) of SPEC2000 benchmarks by simulating each benchmark for 400 million cycles after fast-forwarding an application-specific number of cycles. Based on the IPC, we categorize the 15 benchmarks as high, medium, and low IPC. This classification is shown in Table II. The applications are dynamically simulated, generating a checkpoint after each iteration. The checkpoints for the remaining applications in the SPEC suite were very large, which resulted in extremely long simulation times. Hence, they are omitted from our analysis.

We created six different assignments of applications onto cores based on their IPC characteristics. The assignments are a mix of only high IPC (*h*-*h*), high and medium IPC (*h*-*m*), high and low IPC (*h*-*l*), only medium IPC (*m*-*m*), medium and low IPC (*m*-*l*), and only low IPC (*l*-*l*) benchmarks. The assignment of applications onto 16 cores for each set is shown in Table III. We refer to these collections of application sets as the *training* set (shown in columns 2 through 7 of Table III). This training set has been profiled to form the thermal maps, which were then used for the nonuniform clustering-based placement. We also created another assignment of applications onto 16 cores using 4 different benchmarks (not included in our initial 15 benchmark set) and we refer to it as the *test* set (shown in the rightmost column of Table III). We have used the *test* set to verify the effectiveness of the nonuniform clustering-based placement technique.

Table III. Assignment of Benchmark Sets onto 16 Different Cores

Core	h-h	h-m	h-l	m-m	m-l	l-l	Test set
1	apsi	bzip2	ammp	gap	Gap	applu	parser
2	perlbmk	gap	applu	sixtrack	sixtrack	equake	wupwise
3	gzip	sixtrack	equake	facerec	facerec	art	eon
4	galgel	facerec	art	mgrid	mgrid	mcf	twolf
5	galgel	mgrid	mcf	gap	ammp	equake	wupwise
6	mesa	galgel	mesa-	bzip2	bzip2	ammp	eon
7	gzip	mesa	galgel	sixtrack	applu	applu	wupwise
8	mesa	gzip	gzip	bzip2	equake	ammp	twolf
9	mesa	perlbmk	perlbmk	bzip2	Gap	ammp	parser
10	apsi	mesa	apsi	facerec	sixtrack	art	eon
11	mesa	apsi	mesa	bzip2	bzip2	ammp	eon
12	perlbmk	galgel	galgel	mgrid	Mcf	mcf	wupwise
13	galgel	gzip	gzip	gap-ref	ammp	applu	twolf
14	gzip	bzip2	ammp	sixtrack-ref	applu	equake	parser
15	perlbmk	gap	applu	facerec	equake	art	parser
16	apsi	sixtrack	equake	mgrid	art	mcf	twolf

We run M5 in SE mode and start our simulation with statically assigned benchmarks onto 16 cores. The simulation is first fast-forwarded by an application-specific number of instructions as proposed by Sherwood et al. [2001]. The applications are then simulated for 15 iterations, each consisting of 125 million cycles. To obtain architectural level dynamic power data, we have integrated Wattch infrastructure [Brooks et al. 2000] with 90-nanometer power model into M5.

Thermal simulation is performed using HotSpot version 3.1 [Skadron et al. 2003]. The floorplan of each core is the same as that of a 90-nm technology Alpha 21364 without the level-2 cache (the way level-2 cache is modeled depends on whether we simulate the dense or the sparse architecture), which has a size of 0.43 cm $\times$ 0.43 cm. This floorplan is depicted in Figure 5. The power dissipation of the processor blocks from M5 and the floorplan is used as inputs to HotSpot. We performed the thermal simulation on both CMP floorplans shown in Figure 3. Figure 10 redraws the two floorplans and gives the relative size of the building blocks, where L is equal to 0.43 cm, the side length of an Alpha core. As indicated by Figure 10, the dies for the dense and sparse architecture have the sizes of 2.58 cm $\times$ 2.58 cm, and 2.795 cm $\times$ 2.795 cm, respectively. Our spreader and sink are both made of copper. The spreader is 0.1 cm thick and 6 cm $\times$ 6 cm, and the sink has a base of 0.7 cm and 12 cm $\times$ 12 cm. The thickness of the die and the thermal interface material are set to 0.05 and 0.0075 cm, respectively. For other thermal simulation parameters, we use the default settings of HotSpot. Between the two alternative thermal simulation options of HotSpot, we have used the grid mode thermal simulation, which suits our goal better. In this 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, but increases the simulation

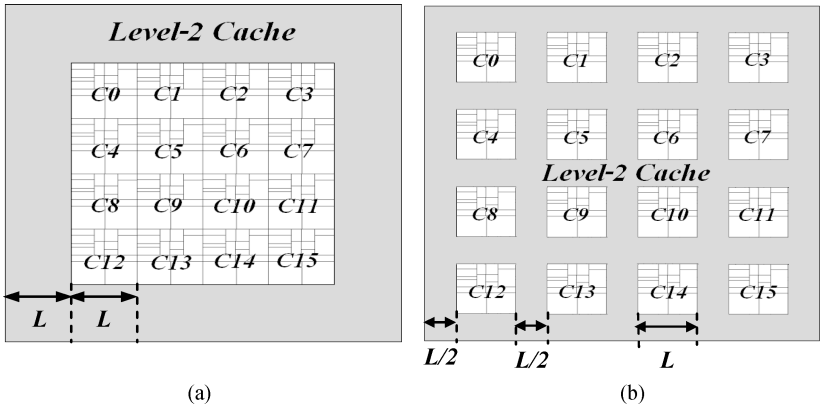

Fig. 10. Floorplans of the CMP chip for (a) the dense architecture with L2 cache surrounding the array of cores (b) the sparse architecture with L2 cache distributed between the cores.

time. We have used a grid size of 288 $\times$ 288 for the entire chip for the dense architecture and 312 $\times$ 312 for that of the sparse architecture, both corresponding to a grid size of 48 $\times$ 48 per core. We first perform the steady-state thermal analysis (grid mode) and use the steady-state temperature profile of the chip as the initial temperature for transient analysis. This represents the die temperature if the processor was already executing instructions prior to execution of benchmarks to model the warm-up period. The ambient temperature is set to 40°C. Our goal is to capture the transient thermal behavior at any point in the execution, so the simulations are divided into intervals that correspond to 0.05 s of execution. For a 2.5-GHz clock frequency, this corresponds to a sampling period of every 125 million cycles, which dictated setting the *simulation interval* to 125 million cycles. The entire thermal simulation lasts for 15 simulation intervals (1.875 billion cycles), which is translated to 0.75 s of CPU execution time. The initial temperature at the start of each interval is taken from the final temperature of the previous iteration. When the maximum temperature of a core is above the emergency temperature (which is set to 82°C in our experiment), we will throttle the core using dynamic voltage and frequency scaling, as explained in Section 3. The dynamic power output from M5 will be scaled accordingly. Based on the scaled dynamic power, we calculate the temperature and the leakage power using iterative method until the leakage power converges.

6.2 Evaluation of Different Sensor Placement Schemes

In evaluating the quality of different sensor placement schemes, we focus on the hotspots whose temperatures are above the emergency temperature level (82°C). Those hotspots with lower temperatures are not taken into account, as they do not pose any threat on system timing/reliability. We assess the sensor placement methods from two different aspects.

First, for each benchmark set and each particular sensor placement, we measure the sensor errors. We use the following definitions for our evaluation:

$T^{am}_{i,j}$: the actual maximum temperature of the ith core during the j th simulation interval.

$T^{sm}_{i,j}$: the maximum value among the temperature readings captured by the sensors on the ith core during the j th simulation interval.

Reading error: $E_{ij} = T^{am}_{i,j} - T^{sm}_{i,j}$, i.e., the difference between the actual maximum temperature and the maximum value among the temperature readings captured by the sensors on the ith core during the j th simulation interval.

Maximum error: $E_{\max} = \max_{i,j}\{E_{ij} | T^{am}_{i,j} \geq 82^{\circ}\text{C}\}$, i.e., the maximum of the reading errors over all the cores and all simulation intervals. Notice here that we only account for the reading error for the hotspots whose temperatures are above 82°C.

Average error: $E_{avg} = \sum_{i,j} \alpha_{ij} E_{ij} / \sum_{i,j} \alpha_{ij}$, i.e., the average of the reading errors over all the cores and all simulation intervals. Here α_{ij} signifies whether the actual maximum temperature of the ith core during the j th simulation interval is larger then 82°C, i.e., $\alpha_{ij} = 1$ if $T^{am}_{i,j} \geq 82^{\circ}$C, otherwise, $\alpha_{ij} = 0$.

Standard deviation of the errors: $E_{std} = \sqrt{(\sum_{i,j} \alpha_{ij}(E_{ij} - E_{avg})^2)/\sum_{i,j} \alpha_{ij}}$, i.e., the standard deviation of the reading errors over all cores and all simulation intervals.

Although these error parameters are related to the quality of different sensor placement schemes, they do not directly reflect the impacts on the hotspots on system reliability and performance. To attain a better understanding of the sensor placement schemes, for each particular sensor placement, we also examine its *thermal emergency and crisis coverage* for each benchmark set. We use the following terminology for this evaluation:

Thermal emergency: the actual maximum temperature of a core is above the emergency temperature level (82°C).

Thermal crisis: the actual maximum temperature of a core is above the *crisis temperature level* (which is set to 90°C in our experiments).

Missed emergency: the actual maximum temperature of a core has reached 82°C, but the estimation of maximum temperature given by the on-core sensors is still below 82°C

Missed crisis: the actual maximum temperature of a core has reached 90°C, but the estimation of maximum temperature given by the on-core sensors is still below 82°C.

Fake emergency: the actual temperature of the hotspot of a core has not yet reached 82°C, but the estimation of maximum temperature given by the on-core sensors is above 82°C.

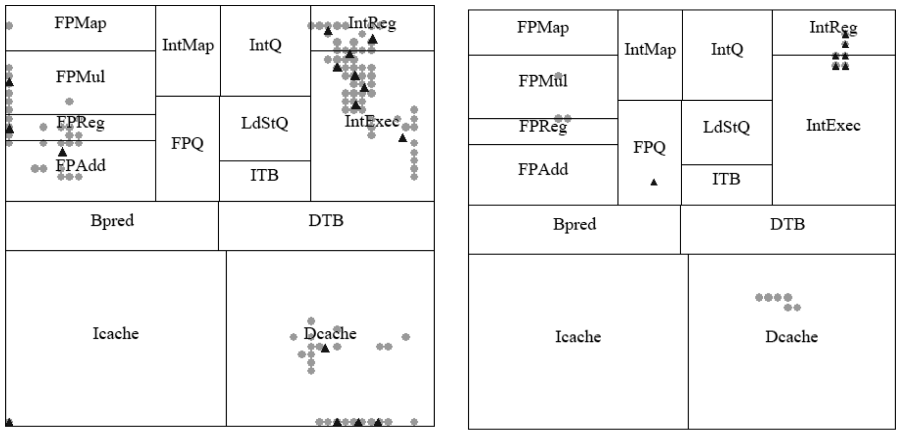

Fig. 11. K-means sensor locations for the dense (left) and the sparse architecture (right).

We count the numbers of missed emergencies and missed crises for each particular sensor placement for each benchmark set. Specifically, we count the number of fake emergencies for the interpolation scheme, since the interpolation method sometimes overestimates the hotspot temperature, incurring system performance penalty. These parameters reflect the emergency and crisis coverage of different sensor placement techniques.

In this section, we will first provide the results for the sensor error measurement. We will then, present the results on emergency and crisis coverage for different sensor placement methodologies.

6.2.1 *Results for Sensor Error Measurement.* First, we present the results for sensor placement by thermal-aware K-means clustering of the hotspots. As mentioned in Section 4, the hotspots for each core at every iteration across the six application sets (the *training* set) form the data points for the clustering. We place 4, 16, and 36, for both architectures by applying thermal-aware K-means. These placements will be referred to as *cluster-4*, *cluster-16*, and *cluster-36*, respectively. All such hotspot locations are folded into a single core. The resulting maps are shown in Figures 5 and 11 for both architectures. Figure 11 also depicts the sensor placement for *cluster-16*. The hotspots are marked with dots, while the sensors are represented by triangles. To assess the "goodness" of the clustering method, we set the placement of the sensors as dictated by the training set and then compute the maximum and average error in reading for the training and test sets based on this placement. Figure 12 shows the maximum, errors, and standard deviation of the errors for the seven benchmark sets, where notation E_{max}, E_{avg}, and E_{std} are short for maximum, average, and the standard deviation of the errors, respectively. Notice that for the sparse architecture, the error values for benchmark set l-l are all zeros. This is because of the fact that the hotspot temperatures never exceed 82°C. Therefore, none of the sensor reading errors of this benchmark set was actually taken into account in our evaluation. The clustering algorithm is efficient for the sparse architecture. For *cluster-16*, the mean of maximum errors across the seven sets is approximately 0.1°C. The largest error observed is less than 0.2°C.

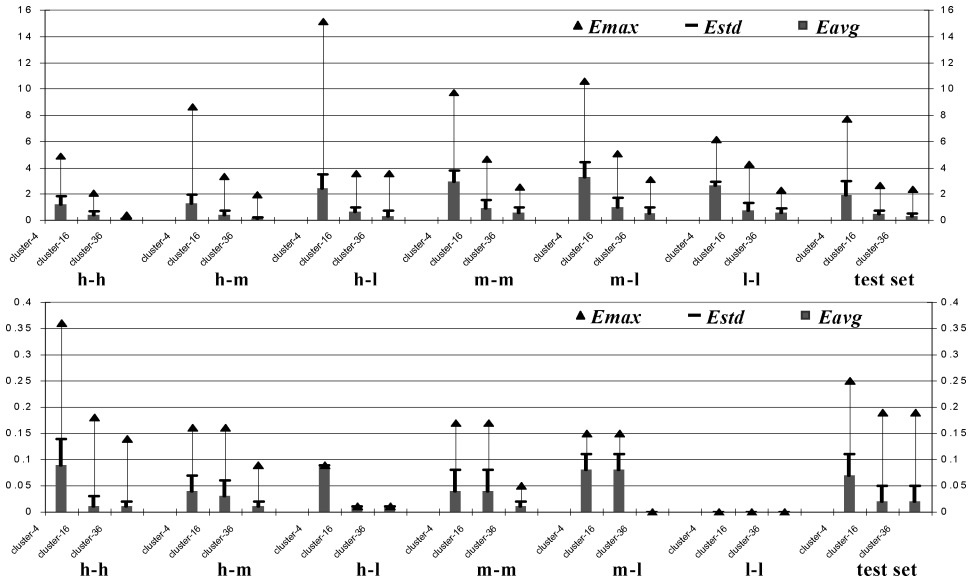

Fig. 12. Maximum and average error for K-means-based clustering method for the dense architecture (upper) and the sparse architecture (lower).

In addition, it is noticeable in Figure 12 that the maximum and average errors of the dense architecture are much larger than those of the sparse architecture. For instance, for the dense architecture, when there are 16 sensors present on each core, the average of the maximum errors across all the benchmark sets is 3.8°C. This average value for the sparse architecture is 0.1°C. Only after employing 36 sensors per core can the thermal-aware K-means clustering achieve accurate results on the dense architectures. As we mentioned in Section 3, on the floorplan of the dense architecture, the cores are next to one another creating a heavy thermal coupling on each other. A hotspot could either be created by a computation intensive component within the same core or by lateral heat flow from an adjacent core. As Figure 5(a) shows, this results in dispersed distribution of hotspots, which is a disadvantage for any profile-driven mechanism. Note that our results are consistent with those of Mukherjee and Memik [2006], where the thermal-aware K-means algorithm is shown achieving good results for single-core sensor placement. In other words, smart placement algorithms can be effectively used for sparse architectures that exhibit little thermal interaction between different cores. However, as we have seen, for the floorplan styles where thermal interaction between cores is strong, such intelligent placement may not be a good choice.

Our second set of results is shown in Figure 13. It represents the errors in sensor readings for each core for uniform grid-based sensor placement. We have experimented with different strategies, equally dividing a core into quadrants and placing a sensor at the center of each quadrant: four quadrants (*grid-4*), sixteen quadrants (*grid-16*), and thirty-six quadrants (*grid-36*). We observe that blindly allocating sensors on grid points and using their maximum as the

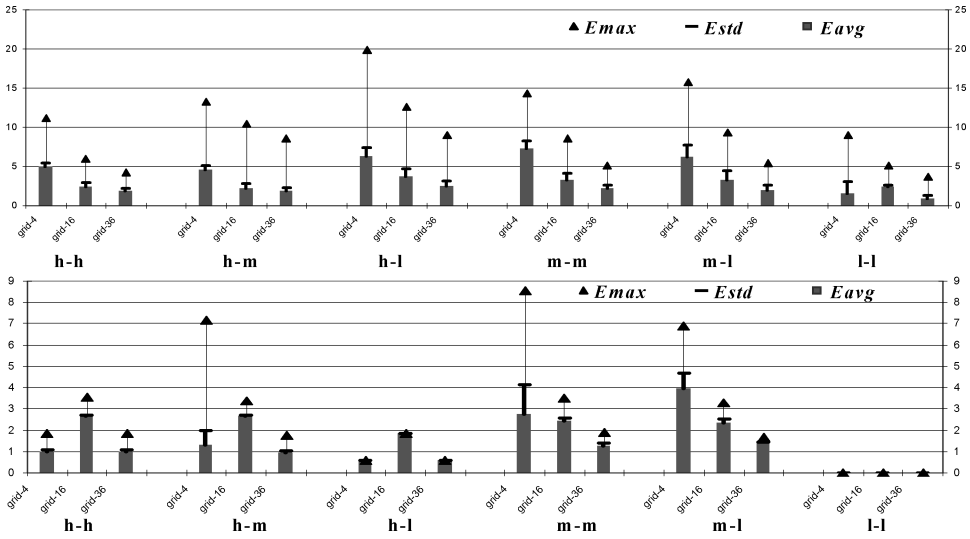

Fig. 13. Maximum and average error for uniform grid-based placement for the dense architecture (upper) and the sparse architecture (lower).

estimation of hotspot temperature, could lead to large error. For example, with 16 sensors on each core, the maximum sensor reading error can be large as 12.6°C. Even as we increase the sensor number to 36 per core, the maximum reading error can still be 9.0°C. Especially for the dense architecture, for each benchmark set, the maximum reading error are above or at least close to 5°C. For benchmark set h-m and h-l, the maximum reading error can even achieve 10°C with 36 sensors on each core.

Next, we present the results for interpolation methods in Figure 14. We have performed interpolation on grids of sizes 16 and 36, which we refer to as *intp-16* and *intp-36*. We also present the errors of *grid-16* and *grid-36* as a comparison. We observe a reduction in the maximum, average, as well as the standard deviation of the errors when using the interpolation with the same number of sensors compared to uniform placement. For example, when using 16 on-core sensors, for the dense architecture, the maximum value of the maximum errors across all the benchmark sets reduces from 12.6 (*grid-16*) to 3.1°C (*intp-16*). For the sparse architecture, using the same number of on-core sensors, the maximum value of the maximum error is reduced from 3.6 (*grid-16*) to 2.3°C (*intp-16*). Also, for *intp-16*, the maximum and average errors have been smaller than those of *grid-36*, in most cases.

Note that, the errors observed are always underestimations for the uniform grid-based method. In some cases, however, we observe that the interpolation method overestimates the absolute maximum temperatures. The errors for the interpolation scheme are computed based on absolute values of these errors capturing both under- and overestimations.

We observe that using the interpolation method, when employing the same number of sensors per core, the error values do not change much as we switch from one architecture to the other. For instance, for the dense architecture,

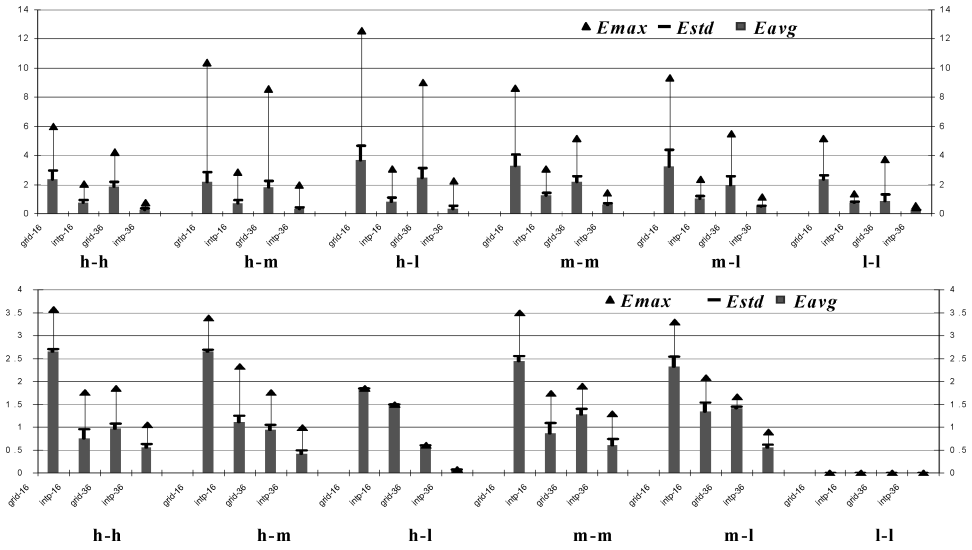

Fig. 14. Maximum and average error of interpolation methods for the dense architecture (upper) and the sparse architecture (lower).

when there are 16 sensors present on each core, the average of the maximum errors across all the benchmark sets is 2.5°C. This average value for the sparse architecture is 1.6°C. As a uniform sensor placement method, the deployment of the sensors in the interpolation scheme does not depend on the thermal profiles. Therefore, the interpolation method enjoys a higher level of "portability" compared to K-means-based clustering method.

We have experimented with our dynamic sensor selection method on hierarchical sensor grids deploying 16 deployed sensors (4 coarse-grain) and 36 deployed sensors (9 coarse-grain). These schemes are referred to as dyn-16–4 and dyn-36–9, respectively.

For *dyn-16–4*, 13 sensors are invoked (4 coarse-grain sensors activated initially and 9 fine-grain sensors activated subsequently), resulting in a 18.75% reduction on total sensor data to be routed and processed. For *dyn-36–9*, 18 sensors are invoked (9 coarse-grain sensors plus 9 fine-grain sensors). The reduction on sensor data volume in this case is 50%. The experimental results are shown in Figure 15, in which the errors of *grid-16* and *grid-36* are also depicted for comparison. It is clear that for both architectures across all benchmark sets, interpolation-based dynamic selection scheme yields almost the same accuracy compared to grid-based sensor placement, while the number of activated sensor is smaller. Note that no improvement in accuracy can be expected using the dynamic selection scheme over grid-based scheme when same number of physical sensors are placed on the chip. In this sense, dynamic selection scheme achieves the accuracy limit bounded by the grid-based while effectively reducing the data volume needed to be communicated and processed.

6.2.2 *Results for Emergency and Crisis Coverage.* Table IV provides the results for the K-means-based clustering method. The columns under "Emer" and

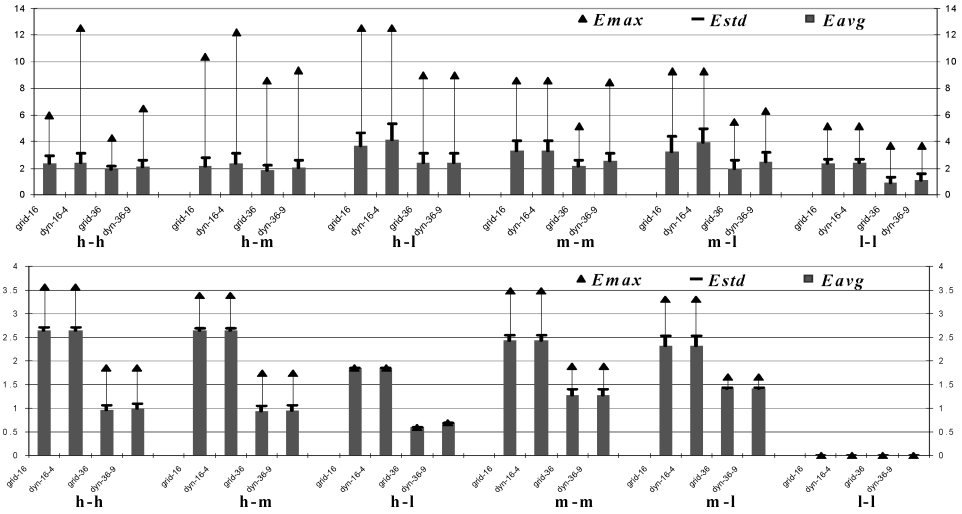

Fig. 15. Maximum and average error of interpolation-based dynamic selection scheme for the dense architecture (upper) and the sparse architecture (lower).

"Crisis" give the number of thermal emergencies and crises for each benchmark set during the execution, while the columns under entry "Missed Emer" and "Missed Crisis" specify the number of missed emergencies and missed crises. For the dense (or sparse) architecture, the total number of emergencies and crises are 667 and 544 (or 402 and 233), respectively. We observe that after employing 16 or more sensors per core, the sensor system will be able to capture all the crises. In the meanwhile, the total number of missed emergencies for all benchmark sets is 3.6% (24 over 667) over the total number of emergencies. Especially for the sparse architecture, with only four sensors per core, all except one thermal emergency are captured. However, note that K-means-based method does not perform as well on the dense architecture as on the sparse architecture. This result conforms to our analysis in Section 6.2.1, where we pointed out that profiling data-driven sensor placement methods are not very effective in monitoring sparsely distributed hotspots.

Next, we present the results for the grid-based sensor placement in Table V. We observe that even as we increase the sensor number to 16 per core, the thermal monitoring infrastructure cannot cover all the thermal crises (benchmark set m-l on the dense architecture). Also, 10.6% (71 over 667) of the emergencies cannot be captured using the same number of on-chip sensors. Even after inserting 36 grid sensors in each core, the percentage of the missed emergencies can still be 6.9% (46 over 667). For instance, on the dense architecture, for benchmark set h-l, with 36 embedded sensors on each core, there are still 22 emergencies that could not be captured.

Table VI shows the results for the interpolation scheme. As we have mentioned in Section 6.1, the interpolation scheme sometimes overestimates the hotspot temperature, resulting in spurious alerts that lead to unnecessary throttling. Therefore, besides the numbers of missed emergencies and crises

Table IV. K-Means Based Sensor Placement Results

	Dense Architecture							Sparse Architecture						
		Missed Emer				Missed Crisis			Missed Emer				Missed Crisis	
	Emer	Cluster 4	Cluster 16	Cluster 36	Crisis	Cluster 4	Cluster 16	Cluster 36	Emer	Cluster 4	Cluster 16	Cluster 36	Crisis	Cluster 4	Cluster 16	Cluster 36
h-h	112	0	0	0	112	0	0	0	107	1	0	0	83	0	0	0
h-m	232	8	4	0	119	0	0	0	75	0	0	0	22	0	0	0
h-l	101	21	5	5	58	0	0	0	1	0	0	0	0	0	0	0
m-m	112	0	0	0	112	0	0	0	94	0	0	0	70	0	0	0
m-l	74	12	7	4	31	2	0	0	13	0	0	0	7	0	0	0
l-l	36	14	8	8	0	0	0	0	0	0	0	0	0	0	0	0
t-s	112	0	0	0	112	0	0	0	112	0	0	0	51	0	0	0

Table V. Grid-Based Sensor Placement Results

	Dense Architecture								Sparse Architecture							
		Missed Emer				Missed Crisis				Missed Emer				Missed Crisis		
	Emer	Grid-4	Grid-16	Grid-36	Crisis	Grid-4	Grid-16	Grid-36	Emer	Grid-4	Grid-16	Grid-36	Crisis	Grid-4	Grid-16	Grid-36
h-h	112	0	0	0	112	0	0	0	107	2	7	2	83	0	0	0
h-m	232	38	15	9	119	1	0	0	75	8	17	4	22	0	0	0
h-l	101	34	28	22	58	5	0	0	1	0	1	0	0	0	0	0
m-m	112	1	0	0	112	1	0	0	94	0	0	0	70	0	0	0
m-l	74	32	12	7	31	10	2	0	13	6	2	3	7	0	0	0
l-l	36	10	16	8	0	0	0	0	0	0	0	0	0	0	0	0

Table VI. Interpolation Scheme Results

	Dense Architecture							Sparse Architecture						
		Intp-4	Intp-16	Intp-36		Intp-4	Intp-16	Intp-36		Intp-4	Intp-16	Intp-36		Intp-4	Intp-16	Intp-36
	Emer	Missed/Fake Emer			Crisis	Missed Crisis			Emer	Missed/Fake Emer			Crisis	Missed Crisis		
h-h	112	0/0	0/0	0/0	112	0	0	0	107	0/3	4/0	0/0	83	0	0	0
h-m	232	3/1	0/2	0/0	119	0	0	0	75	1/2	8/0	1/0	22	0	0	0
h-l	101	7/8	2/2	1/1	58	0	0	0	1	0/2	1/0	0/0	0	0	0	0
m-m	112	0/0	0/0	0/0	112	0	0	0	94	0/0	0/0	0/0	70	0	0	0
m-l	74	3/21	0/2	0/1	31	0	0	0	13	6/3	1/0	1/0	7	0	0	0
l-l	36	4/39	7/0	0/3	0	0	0	0	0	0/0	0/0	0/0	0	0	0	0

in Table VI, we also report the number of times our thermal-monitoring infrastructure initiates fake emergency alerts. As we can see, with only four sensors per core, we have been able to capture all the thermal crises. If we increase the number of sensors to 16 per core, the percentage of missed emergencies is reduced to 1.3% (9 over 667) for the dense architecture and to 2.1% (14 over 667) for the sparse architecture. The total number of fake alerts is only six for the dense architecture and zero for the sparse architecture. If we increase the number of sensors to 36 per core, we will be able to capture almost all the thermal emergencies. Also, we observe that different from the K-means-based clustering method, the interpolation scheme performs well on both architectures, especially after we increase the number of sensors to 16 per core. This result is consistent with our observation on the error measurement results of the same technique discussed in Section 6.2.1.

Finally, we present the experimental results of the dynamic selection scheme in Table VII. As discussed earlier, the quality of this method (with the same number of sensors) is bounded by the grid-based placement method. Our interest only lies in whether we can achieve approximately the same hotspot coverage as the grid-based method, while eliminating a large portion of sensor reading data needed to be routed and processed. Compared to Table V, it is clear that using the dynamic selection scheme, in most of the cases, the numbers of missed emergencies/crises only slightly surpass those of the grid-based scheme, revealing that our dynamic selection scheme could, on one hand, eliminate a large percentage of on-chip sensors needed to be activated and, on the other hand, still approximate the accuracy of the grid-based placement using the same number of sensors.

6.3 Discussion

In summary, our experimental results show that thermal-aware K-means sensor placement strategy, though being efficient for single core and some CMP architectures, may not be suitable for all CMP chips, especially for those having strong intercore thermal interaction. We note that even for the architectures where thermal-aware K-means algorithm effectively minimizes the reading errors and the number of missed thermal emergencies, the sensor placement still depends on the benchmark sets. The nonuniform placement technique using clustering relies on profiling data and resulting thermal maps of hotspots. Although the benchmark suite we have used aims to represent a fairly typical workload for a high-performance microprocessor, there can be unexpectedly large variations in some outlier applications, which could not be captured by the benchmark suite. Also, new applications emerge over time. To address these problems, we first examined the grid-based sensor placement. Our results show that simply placing sensors in a grid can present significant inaccuracies even when the number of sensors is increased to 36 per core. Our interpolation method reduces the amount of sensors to a great extent while maintaining relatively small average reading errors. For example, for the dense architecture, the maximum value of the maximum errors across all the benchmark sets reduces from 12.6 (*grid-16*) to 3.1°C (*intp-16*), which is even much smaller than

Table VII. Dynamic Selection Scheme Results

	Dense Architecture						Sparse Architecture					
		Dyn-16-4	Dyn-36-9		Dyn-16-4	Dyn-36-9		Dyn-16-4	Dyn-36-9		Dyn-16-4	Dyn-36-9
	Emer	Missed Emer		Crisis	Missed Crisis		Emer	Missed Emer		Crisis	Missed Crisis	
h-h	112	1	0	112	1	0	107	7	3	83	0	0
h-m	232	17	16	119	0	0	75	17	4	22	0	0
h-l	101	32	22	58	4	0	1	1	1	0	0	0
m-m	112	0	0	112	0	0	94	0	0	70	0	0
m-l	74	19	13	31	2	0	13	2	3	7	0	0
l-l	36	16	11	0	0	0	0	0	0	0	0	0

the 9.0°C error observed with *grid-36*. Also, in terms of thermal emergency coverage, the percentage of uncovered emergencies is reduced from 10.6 (*grid-16*) to 1.3% (*intp-16*). It is intriguing that although Lee et al. [2005] have pointed out the difficulty of finding an exact equation to fit the temperature-distance curve, our experiments reveal that simple interpolation scheme has achieved acceptable accuracy. Furthermore, dynamic selection scheme results in similar accuracy with large reduction in number of sensors involved in data collection over the grid-based scheme. For both architectures, when only invoking 18 out of 36 sensors (*dyn-36–9*), dynamic selection scheme yields almost the same accuracy and emergency coverage as *grid-36*. The interpolation and dynamic selection mechanisms are built upon a static placement, independent of the profiling assumptions. The interpolation scheme achieves relatively small reading errors using a small number of sensors; the dynamic selection method effectively reduces the amounts of sensor data routed at a time and has reasonable accuracy.

7. CONCLUSIONS

We have introduced novel techniques to estimate the maximum temperature on a multicore chip. Our goal is to provide accurate temperature readings on a given chip while maintaining a reasonable overhead in terms of sensor data collection. Dynamic thermal management schemes can leverage on the sensor infrastructure that are built by our systematic approaches.

We first experiment with the thermal-aware K-means algorithm, which has been shown to be successful for single-core sensor placement. Our results revealed that this method might not be suitable for some CMP architectures. We then analyzed a grid-based sensor placement and observed that the number of sensors should be increased dramatically to achieve high accuracy. We improved upon this solution by introducing (1) an interpolation scheme and (2) a dynamic selection scheme. The interpolation scheme is able to reduce the average errors for a given distribution. On the other hand, the dynamic selection method provides reasonable accuracy, by using only a relatively small fraction of embedded sensors per core. Overall, these schemes achieve accurate temperature readings with small number of embedded sensor readings—a desirable property for multicore processors.

REFERENCES

BINKERT, N. L. D. ET AL. 2006. The M5 simulator: Modeling networked systems. *IEEE Micro*, *26*, 4, 52–60.

BLACKBURN, D. L. 2004. Temperature measurements of semiconductor devices – A Review. In *Semiconductor Thermal Measurement, Modeling and Management Symposium*.

BORKAR, S. 1999. Design challenges of technology scaling. *IEEE Micro* (July-Aug.) *19*, 4, 23–29.

BROOKS, D., TIWARI, V., AND MARTONOSI, M. 2000. Wattch: A framework for architectural-level power analysis and optimizations. In *International Symposium on Computer Architecture*.

BORKAR, S., KARNIK, T., NARENDRA, S., TSCHANZ, J., KESHAVARZI, A., AND DE, V. 2003. Parameter variations and impact on circuits and microarchitecture. In *Design Automation Conference*. Anaheim, CA.

BORKAR, S. ET AL. 2005. Platform 2015: Intel processor and platform evolution for the next decade. Whitepaper.

BRATEK, P. AND KOS, A. 2001. Temperature sensors placement strategy for fault diagnosis in integrated circuits. In *Symposium on Semiconductor Thermal Measurement and Management*.

CONG, J., WEI, J., AND ZHANG, Y. 2004. A thermal-driven floorplanning algorithm for 3D ICs. In *International Conference on Computer-Aided Design*.

DORSEY, J. ET AL. 2007. An integrated quad-core opteron processor. In *International Solid-State Circuits Conference*.

GUNTHER, S. ET AL. 2001. Managing the impact of increasing microprocessor power consumption. *Intel Tech. J.*

http://www.capgo.com/Resources/Temperature/Semiconductor/Semi.html. Introduction to Semiconductor Temperature Sensors.

http://download.intel.com/design/Pentium4/datashts/29864312.pdf. Intel Pentium 4 Processor with 512-KB L2 Cache on 0.13 Micron Process Thermal Design Guidelines. 2002.

http://www.maxim-ic.com/appnotes.cfm?an_pk=689. IC Temperature Sensor Find the Hot Spots. 2001.

HUANG, W. ET AL. 2004. Compact thermal modeling for temperature-aware design. In *Design Automation Conference*.

HUANG, W. ET AL. 2000. A framework for dynamic energy efficiency and temperature management. In *International Symposium on Microarchitecture*.

JUANG, P. ET AL. 2005. Coordinated, distributed, formal energy management of chip multiprocessors. In *International Symposium on Low Power Electronics and Systems*.

KAXIRAS, S. AND XEKALAKIS, P. 2004. 4T-decay sensors: A new class of small, fast, robust, and low-power, temperature/leakage sensors. In *International Symposium on Low Power Electronics and Design*.

KONGETIRA, P., AINGARAN, K., AND OLUKOTUN, K. 2005. Niagara: A 32-way multithreaded sparc processor. *IEEE Micro*, *25*, 2, 21–29.

KRINITSIN, V. *Pentium 4 and Athlon XP: Thermal Conditions*. Available from http://www.digit-life.com/articles/pentium4athlonxpthermalmanagement/.

KUMAR, R., ZYUBAN, V., AND TULLSEN, D. M. 2005. Interconnections in multi-core architectures: understanding mechanisms, overheads and scaling. In *International Symposium on Computer Architecture*.

LEE, K.-J., SKADRON, K., AND HUANG, W. 2005. Analytical model for sensor placement on microprocessors. In *International Conference on Computer Design*.

LI, Y., BROOKS, D., BROOKS, HU, Z., SKADRON, K. 2005. Performance, energy, and thermal considerations for SMT and CMP architectures. In *Proceedings of the 11th International Symposium on High-Performance Computer Architecture*. San Francisco, CA.

LIU, Y. ET AL. 2007. Accurate Temperature-dependent integrated circuit leakage power estimation is easy. In *Design, Automation and Test in Europe*.

LOPEZ-BUEDO, S., GARRIDO, J., AND BOEMO, E. I. 2002. Dynamically inserting, operating, and eliminating thermal sensors of FPGA-based systems. *IEEE Trans. Components Packaging Tech. 25*, 4, 561–566.

MONDAL, S., MUKHERJEE, R., AND MEMIK, S. O. 2006. Fine-grain thermal profiling and sensor insertion for FPGAs. In *IEEE International Symposium on Circuits and Systems*.

MUKHERJEE, R. AND MEMIK O. S. 2006. Systematic temperature sensor allocation and placement for microprocessors. In *IEEE/ACM Design Automation Conference (DAC)*. San Francisco, CA.

MUKHERJEE, R., MONDAL, S., AND MEMIK, S. O. 2006. Thermal sensor allocation and placement for reconfigurable systems. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. San Jose, CA.

PHAM, D. ET AL. 2005. The design and implementation of a first generation cell processor. In *International Solid-State Circuits Conference*.

POWELL, M. D., GOMMA, M., AND VIJAYKUMAR, T. N. 2004. Heat-and-Run: Leveraging SMT and CMP to manage power density through the operating system. In *International Conference on Architectural Support for Programming Languages and Operating Systems*.

POIRIER, C. ET AL. 2005. Power and temperature control on a 90nm itanium-family processor. In *International Solid-State Circuits Conference*.

QUENOT, G. M., PARIS, N., AND ZAVIDOVIQUE, B. 1991. A Temperature and voltage measurement cell for VLSI circuits. In *Euro ASIC Conference*.

RATTNER, J. R. 2005. Keynote at the intel developer conference. Available from http://www.intel.com/technology/techresearch/idf/platform-2015-keynote.htm.

ROTEM, E. ET AL. 2004. Analysis of thermal monitor features of the Intel Pentium M processor. In *Workshop on Temperature-aware Computer Systems*.

ROTEM, E. ET AL. 2006. Temperature measurement in the Intel core duo processor. In *International Workshop on Thermal Investigations of ICs*.

SANKARANARAYANAN, K. ET AL. 2005. A case for thermal-aware floorplanning at the microarchitectural level. *J. Instruction-Level Parallelism*. *7*, 1–16.

SHERWOOD, T., PERELMAN, E., AND CALDER, B. 2001. Basic block distribution analysis to find periodic behavior and simulation points in applications. In *International Conference on Parallel Architectures and Compilation Techniques*.

SKADRON, K. ET AL. 2003. Temperature-aware microarchitecture. In *International Symposium on Computer Architecture*.

SPEC-CPU2000. 2000. *Standard Performance Evaluation Council, Performance Evaluation in the New Millennium, Version 1.1.*

SRINIVASAN, J. ET AL. 2004. The case for lifetime reliability-aware microprocessors. In *International Symposium on Computer Architecture*.

TUTHILL, M. 1998. A switched-current, switched-capacitor temperature sensor in 0.6 um CMOS. *IEEE J. Solid-State Circuits*. *33*, 7, 1117–1122.

TSAI, J., CHEN, C. C., CHEN, G., GOPLEN, B., QIAN, H., ZHAN, Y., KANG, S., WONG, M. D. F., AND SAPATNEKAR, S. S. 2006. Temperature-aware placement for SOCs. In *Proceedings of the IEEE*. *94*, 8 (Aug.), 1502–1518.

VELUSAMY, S. ET AL. 2005. Monitoring temperature in FPGA based SoCs. In *International Conference on Computer Design*.

WANG, N., ZHANG, S., AND ZHOU, R. 2003. A novel built-in CMOS sensor for on-line thermal monitoring of VLSI circuits. In *International Conference on ASIC*.

Received March 2007; revised August 2007 and January 2007; accepted January 2007