Transient Thermal and Load Aware Distributed Thermal Managment for Many-core Systems

Abstract— **In this paper, we propose a fully distributed thermal management method to reduce the thermal hot spots and on-chip temperature variance, which could lead to better thermal reliability and reduced package costs of future many-core processors without using a central control agent. The proposed method consists of two control knobs: distributed task scheduling and mode switching control logic. The distributed task scheduling method systematically incorporates the load influence from local and adjacent areas, as well as transient thermal effect due to the package related heat conduction to optimize the task load of the microprocessor cores in a distributed way. The mode switching control logic switches the chip between low power mode and normal working mode based on the average temperature as a global thermal indicator, which is tracked and monitored locally in each core, using a newly proposed distributed state tracking technique.**

Our experimental results on a 100-core microprocessor shows that the proposed method works more effectively in reducing the on-chip temperature variance and thermal hot spots comparing with the existing distributed method (42% more thermal hot spot reduction) and the alternative method that does not consider the transient thermal effect (33% more thermal hot spot reduction). Also, it shows that the average temperature could be accurately estimated in a distributed way, which is used by mode switching control logic as an overall thermal indicator to switch the processor between normal mode and low power mode.

I. Introduction

Elevated on-chip temperature has become a top concern for high-performance microprocessor design as more devices are integrated on a chip, and thermal constraints are currently the major driving force for wide adoption of multi/many core architectures. As the power density continues to increase, the excessively high temperature spots, called *thermal hot spot*, would adversely lead to performance degradation, increased cooling costs, reduced chip reliability and accelerated aging of multi-core microprocessors [1]–[4]. Thus, thermal management and related cooling techniques have been identified by ITRS [5] as one of the five key challenges during the next decade of achieving the projected performance goals of the industry.

As part of thermal management method, task scheduling/migration technique serves to balance the temperature of the cores [3], [6], [7]. The general idea of these methods is to migrate the heavy loaded task away from an overheated core to a cooler core. The traditional approach like [6], typically migrate the heaviest tasks (with largest power consumption) to the coolest cores to balance the temperature profile of all the cores. However, such an intuitive decision may lead to sub-optimal result, because it does not take the influence from the adjacent cores into account. To improve the thermal profile, a recent work [3] proposed a new thermal management approach that uses a temperature incremental factor to count the heat flux from the adjacent cores.

However, most existing approaches [3], [6]–[8] are centralized control approaches, which requires a Centralized Manager (CM) to monitor the temperature and power of each core in the many-core microprocessor, and globally reallocate the resources and schedule the tasks. The centralized control method suffers from two major drawbacks. First, the centralized control method is not scalable in the context of future many core processors with hundreds and even thousands of cores [9], [10]. It will suffers from high computational costs inside the CM, and large volume of monitoring and communication traffics around CM. Second, the centralized control method is not reliable for the many-core systems since failure or even a slight malfunction of the CM may result in disastrous damage to the processor. To mitigate these problems of the centralized control approach, the distributed control approaches have been proposed [11], [12]. In [11], an agent-based distributed power/thermal management method is proposed, using dynamic voltage and frequency scaling technique. The thermal management is done via localized agent negotiations and each agent (or the core it represents) acts independently and is reactive to its neighbors only. In [12], a distributed task migration technique, called *DTB-M*, was proposed. DTB-M monitors temperature and power in each local core, and performs task migration only between two adjacent cores. However, the existing approaches, like DTB-M, still suffer from the lack of theoretical guarantee that the temperature will be evenly distributed, since the overall goal of its task migration is to reduce the average temperature across all cores. However, the effort of lowering average temperature across all cores does not necessarily lead to more balanced temperature distribution, and thus the efficiency of reducing the number of thermal hot spots may be compromised.

In this work, we first aim at developing a fully distributed thus scalable task scheduling/migration method to reduce the thermal hot spots and temperature variation of many-core microprocessors. The major contributions of this work are:

1) Unlike the existing distributed approaches, we have defined an additional thermal indicator, named *effective initial temperature*, to consider the behavior of transient thermal effect, and extended it to a distributed manner to provide theoretical support for on-chip thermal balancing.
2) We have introduced two additional distributed migration criteria to consider the load of each local core as well as the influence from its adjacent areas to perform the local task migrations.
3) In order to further reduce the overall temperature of the chip, we have introduced mode switching control logic to the thermal management algorithm by switching the chip between low power mode and normal working

mode based on a newly developed distributed temperature tracking technique. The new distributed temperature tracking technique is able to track the average temperature of the whole chip using local state of each core, and will provide additional global temperature information to each local core in a distributed manner.

Our experimental results on a 100-core microprocessor demonstrate that the distributed using of the additional thermal indicator leads to more effective task migration decision for thermal balancing than directly using temperature value of cores. As a result, the proposed distributed task migration method can reduce the number of the thermal hot spots by 42% compared to the existing distributed method (DTB-M) [12], leading to more uniform on-chip temperature distribution across the microprocessor cores. The results on the low power mode switching control simulation also shows that the average temperature of the chip could be dynamically adjusted with low power mode introduced, and it could be effectively tracked in each local core using the distributed temperature tracking technique.

The rest of the paper is organized as follows. In section II, we present a general outline of the thermal management problem and the motivation to solve it. In section III, we present the proposed method and its innovations with the algorithm flow. In section IV, the proposed method is evaluated and its significance is discussed. Section V concludes this paper.

II. Problem formulation

In this work, task migration scheme is used to reduce the on-chip thermal hot spots for multi-core microprocessors. Specifically, we assume a multi-core microprocessor consists of N tasks of different load, denoted as $\mathbb{P} = [P_1, P_2, \cdots, P_N]$. M processor cores are involved in executing the tasks. Since the load of the tasks are different, the power intensity varies across all the cores, resulting in non-uniformly distributed temperature profile. Using C_p (Unit:J/K) to denote heat capacitance, $P(t)$ to represent the transient power generated in a core, which is related its load, and $\dot{Q}(t)$ to represent the transient heat transfer rate to and from a core, the temperature of a core $T(t)$ could be written as

$$T(t) = \frac{1}{C_p} \int_{t_0}^{t} (P(t) - \dot{Q}(t))dt + T(t_0) \qquad (1)$$

Note that the heat transfer rate $\dot{Q}(t)$ to and from the core could influence the temperature of the core, which suggests that the load in the adjacent region and transient thermal effect of the processor package must be considered when making thermal management decision.

We aim at minimizing the temperature variations by properly distribute the workload power among the cores. That is to minimize **MAX** $T(t)$ for a given time t over the whole chip. Similar to [11], [12], in our distributed control framework, we assume that each core can only talk to its adjacent cores during this task migration process. Each core has the knowledge of the its temperature and power informations of its current task, as well as those of its adjacent cores. As a result, the task migration could only be performed between two adjacent cores based on their local thermal related information, power and load influence from adjacent area.

There are several problems need to be solved. The first problem is how to do the migration properly in a distributed way to deal with the fast changing time-variant temperature on each core. The second problem is to solve the low effectiveness issue of the distributed DTM due to its lack of global temperature infromation compared to the centralized algorithm. The third problem is, in addition to minimize the temperature variations, how to combine additional DTM method to lower the average temperature of the chip. We will discuss and solve these problems in the next section.

III. The new distributed thermal management method

In this section, we present a new fully distributed thermal management techniques. The new technique consists of two knobs to control the temperature: (1) task migration to reduce thermal hot spot and balance the on-chip temperature; (2) mode switching control logic to reduce the chip temperature when the overall temperature exceeds a certain threshold (at cost of degraded performance). To efficiently control the temperature via these two knobs, we adopt several new techniques. First, during the task migration, we look at the transient thermal effects by using a new temperature indicator. Further more, we not only look at the temperature, load of the current core, but also those of immediate neighborhood cores to ensure task exchange will have better chances to reduce temperature variations. Second, a fully distributed control mechanism is employed for task migration and mode switching control logic to achieve better scalability. Unlike the existing approaches use only local thermal information, we use the average temperature among all the cores, which can be computed by a recently proposed distributed state tracking technique, to guide the mode switching control logic, which reduces power and temperature at costs of reduced system throughput.

We remark that the term *fully distributed* has two folds of meanings in this paper: (1) the task migration is controlled in a distributed way, and the migration happens only between two adjacent cores, which is organized in 2D meshed tile structure. (2) the temperature is tracked in a distributed way, and there is no central agent to monitor the global temperature distribution across all the microprocessor cores.

In the following, we first present the overall thermal control flow. Then we introduce the important concepts and steps in the proposed method such as effective initial temperature, distributed state tracking techniques for average temperature calculation and its applications in mode switching control logic, and finally present the proposed distributed task migration scheme.

A. The overall distributed thermal management flow

The overall proposed distributed thermal control flow is summarized in Algorithm 1.

Like existing works such as [12], we also assume that each task occupies an equal slice of execution time (one execution cycle), and power traces for each task could be obtained from OS or predicted from the history of the power traces

Algorithm 1 Proposed distributed thermal management flow

Require: Task loads, many-core processor configuration, T_{th}
Ensure: Optimized on-chip temperature distribution with minimal impacts on performance.
Start program at room temperature
for each execution cycle **do**
1. Obtain power traces under different task loads (via estimation or profiling).
2. Obtain temperature responses of the many-core microprocessor (via thermal sensors or estimation).
3. Estimate average temperature indicator using distributed state tracking algorithm.
if Migration criteria is met **then**
Perform distributed task migration using the proposed scheme in Fig. 5 core by core.
end if
if Estimated average temperature is above T_{th} **then**
mode switching control logic invokes low power mode for all the cores until average temperature below T_{th}.
end if
end for

using some estimation methods like time series prediction methods. We assume that the multi/many-core microprocessor is executing different sets of tasks in different execution cycles as shown in Fig. 1, where P_m and P_n are used to denote the load of the tasks in two adjacent cores.

At the beginning of each execution cycle, the proposed distributed task migration policy is applied to each core, and task migration between two adjacent cores is performed if the task migration criteria (discussed later) is satisfied. In this way, the tasks are scheduled to balance the temperature profile of the multi/many-core microprocessors in the new execution cycle so that the on-chip temperature gradient and occurrence of thermal hot spots during the task execution should be reduced. Also, the distributed state tracking technique is used to dynamically estimate the average temperature at each time step. If the average temperature is within a certain threshold (T_{th}), the processor chip will be working under normal mode with full throughput, otherwise, mode switching control logic will switch the processor to low power mode with reduced working frequency and supply voltage, which reduces the power and temperature at the price of reduced throughput of the system. As a result, low power mode should be invoked only when the system is highly utilized and task migration in this case will not be able to keep the temperature below some threshold temperature. In our work, the mode switching control monitors the computed average temperature as the indicator for invoking low power mode, as average temperature is a good indicator for the system utilization rate.

B. Effective initial temperature for task scheduling

Traditional task migration methods typically assign heavy tasks to the cores with low temperature to balance the on-chip temperature profile [6], [7]. However, such a strategy may not lead to best balanced temperature distributions specially in terms of transient temperature. In other words, the core with relatively lower temperature may not be a good candidate to take on heavier task in the new execution cycle, because some cores might be sensitive to load changes due to the structure and properties of the package, and carelessly assigning heavy tasks to this type of cores might lead to unexpected higher temperature. To avoid such a problem and make better task migration decision, it is important to predict the transient temperature responses when the load has been changed, which is possible if the thermal conductance G and thermal capacitance C of the processor package are considered.

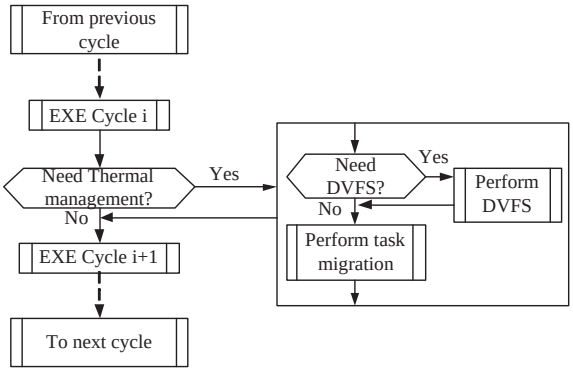


Fig. 1. The run-time thermal management framework

One can always perform full-blown transient thermal simulations to predict the transient temperature responses given task loads of cores. However, such simulation will be very expensive in the control loop. Instead, we apply *effective initial temperature* as a simple yet effective temperature indicator for task migration without full transient simulation.

1) 0th-moment temperature based indicator: Moment matching based analysis has been used for fast estimation of interconnect delays in the past [13]. In this work, we apply moment matching based analysis for fast transient thermal estimation. Instead of performing the full moment matching based analysis, we would like to derive a temperature indicator to guide the task migration. This scheme is motivated by the observation that the 0th moment component of the power traces of the microprocessor is dominant compared to high frequency components. Fig. 2 shows the Fourier transformation of the power trace from a benchmark, which clearly indicates the dominance of the 0th moment component. As a result, the corresponding 0th moment of the transient temperature responses can be used as a good indicator of the temperature for each core.

Mathematically, for equivalent thermal circuit using G as the thermal conductance matrix and C as the thermal capacitance matrix, we can apply Modified Nodal Analysis (MNA) to formulate the thermal circuit

$$GT(t) + C\dot{T}(t) = u(t) \tag{2}$$

where $T(t)$ is the time-domain temperature response, and $u(t)$ is the power trace of the processor.

Assume $U(s)$ as the Laplace transformation of the power trace $u(t)$ of that processor, and $T(s)$ as the corresponding temperature response, in frequency domain, we could have

$$GT(s) + sCT(s) - CT(t_0) = U(s) \tag{3}$$

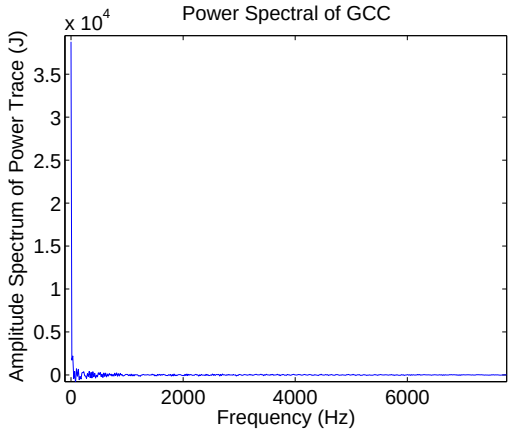


Fig. 2. Discrete Fourier transformation of the power trace for running GCC benchmark

where $T(t_0)$ is the initial temperature vector of all the nodes at time zero, t_0, or alternatively, the temperature vector at the end of the previous simulation cycle.

To find out the contribution of the transient temperature in terms of frequency moments, we expand $T(s)$ and $U(s)$ using Taylor's series at $s = 0$ to have

$$G(T_0 + T_1 s + T_2 s + \cdots) + sC(T_0 + T_1 s + T_2 s + \cdots) \\ -CT(t_0) = U_0 + U_1 s + U_2 s + \cdots \quad (4)$$

Since the 0th moment of the power trace U_0 is dominant, the 0th moment component of the transient temperature response T_0 is also dominant, which is shown below by moment matching in (4) as

$$T_0 = G^{-1}U_0 + G^{-1}CT(t_0) \quad (5)$$

Thus, it makes a good sense to use the 0th moment component of the response temperature (called *0th moment temperature*) as an on-chip temperature indicator. In this way, we could conveniently use (5) to calculate the dominant 0th temperature T_0 given the 0th moment component of power trace U_0 for the future task and the initial temperature $T(t_0)$ at the current time. The 0th moment temperature T_0 contains transient effects (G and C) of the thermal system, which will be explored by the this work.

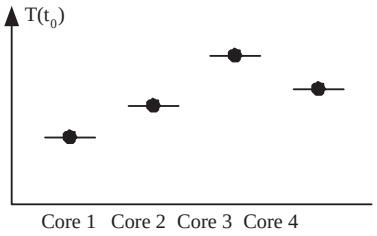
2) Physical insight derived from effective initial temperature: We can define a new term called *effective initial temperature* in frequency domain as

$$T_{eff} = G^{-1}CT(t_0) \quad (6)$$

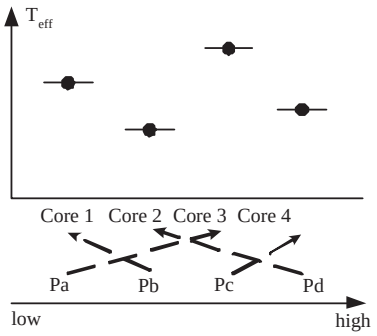
where T_{eff} has temperature unit in frequency domain. As a result, we can rewrite (5) as

$$T = G^{-1}U_0 + T_{eff} \quad (7)$$

Physically, the first term $G^{-1}U_0$ actually represents the steady state temperature responses due to the (predicted or estimated) power of current task cycle. The second term T_{eff} represents the steady state response due to the initial temperature $T(t_0)$ under a specific package structure defined by thermal matrices G and C, which is similar to zero input ($U_0 = 0$) response of the thermal system. With this knowledge, the best way to balance the on-chip temperature distribution is to assign heavy load task to the core with low T_{eff} instead of the core with low $T(t_0)$. Fig. 3 gives an intuitive illustration of the physical meaning of T_{eff} and the task scheduling scheme based on T_{eff} in a 4-core microprocessor, where $T(t_0)$ in Fig. 3 (a) is the initial temperature distribution of the chip package at time t_0, and the effective initial temperature T_{eff} in Fig. 3 (b) represents the steady state response of $G^{-1}C$ system due to $T(t_0)$ under zero inputs. The power of the tasks (Pa, Pb, Pc, Pd) in Fig. 3 (b) are sorted from low to high and assigned to the proper cores to compensate T_{eff} and balance on-chip temperature profile.



(a) The initial temperature distribution at time $T(t_0)$



(b) The resulting temperature distribution T_{eff} under $U_0 = 0$ and the task scheduling scheme to balance the on-chip temperature distribution

Fig. 3. Physical meaning of the effective initial temperature and the task scheduling method based on it

In this way, the transient thermal effect due to the processor package design is considered in addition to the local temperature of the core, which provides theoretical support for on-chip temperature balancing, leading to more effective task scheduling to reduce on-chip hot spots as will be shown in the experiments.

3) Distributed task scheduling based on effective initial temperature: As clarified before, the definition of T_{eff} introduces a new way of task scheduling method for thermal balancing of the processor chip. However, according to (6), the calculation of each element in T_{eff} requires all elements in $T(t_0)$, which is the global distribution of the temperature values across the chip. Thus, we need to estimate a distributed T_{eff} in a distributed way so that it could be used in the

proposed method.

The distributed version of T_{eff} could be obtained in the following way: Instead of using the global temperature distribution and global package model as suggested by (6), we could use a local package model with local temperature values from the current core and its adjacent cores to calculate the distributed effective initial temperature for the local chip region as

$$T_{eff,N(i)} = R_{N(i)} C_{N(i)} T_{N(i)}(t_0) \tag{8}$$

in which $T_{eff,N(i)}$ contains the effective initial temperature values of core i and its adjacent cores in the local region shown by Fig. 4; $T_{N(i)}$ contains the real temperatures of core i and its adjacent cores in the same local region:

$$T_{N(i)} = [T_a, T_b, T_c, T_d, T_e, T_f, T_g, T_i]. \tag{9}$$

$R_{N(i)}$ and $C_{N(i)}$ are the local thermal resistance matrix and thermal capacitance matrix constructed from the elements from row a to i and column a to i from original G^{-1} and C matrices shown in Fig. 4.

We remark that $T_{eff,N(i)}$ is an approximate of T_{eff}. Since temperature rise due to a heat source is a localized effect. Such an approximation by looking at itself and neighbor's contribution gives a good approximation. As a result, the effective initial temperature is calculated distributively in each local region, and could be used in the distributed task migration scheme. The details about the proposed distributed task migration using distributed T_{eff} will be explained later.

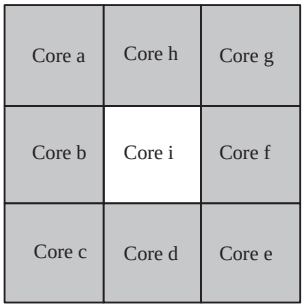


Fig. 4. The local region used to calculate the distributed $T_{eff,N(i)}$ in core i and its adjacent area

C. The proposed new distributed task migration method

In our proposed approach, we use effective initial temperature as a new thermal indicator that takes account of transient thermal effect due to package heat conduction to guide task migration decision. The new approach is distributed in the sense that task migrations happen only between two adjacent cores as shown in Fig. 4, where core a-h are the adjacent cores of core i. To further improve our distributed task migration method, we also consider task load of the cores and the load influence from their adjacent area. The proposed distributed task migration method can be outlined as the following: we define the current core as the one that the task migration policy is applied to, the destined core as the adjacent core that could potentially exchange task with the current core if all of the following task migration criteria are satisfied:

1. $T_{eff,des} < T_{eff,cur}$, where $T_{eff,des}$ is the effective initial temperature of the destined core and $T_{eff,cur}$ is that of the current core.
2. $P_{des} < P_{cur}$, where P_{des} is the task load of the destined core in the new execution cycle, and P_{cur} is the counterpart of the current core.
3. $TNP_{des} < TNP_{cur}$, where TNP_{des} is the total task load of the adjacent cores surrounding the destined cores in the new execution cycle, and TNP_{cur} is the counterpart of the current cores.

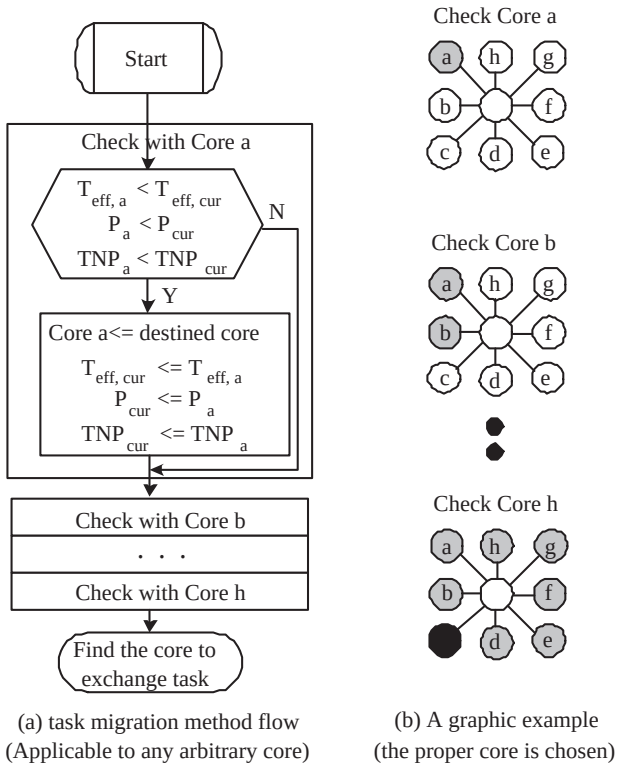
The idea behind such a task migration policy is to ensure that we will likely to have a gain in terms of reduced temperature variance across the cores after the task exchange.

The details of the distributed task migration method flow can be illustrated in Fig. 5 (a). The current core communicates with its adjacent cores to check if the migration criteria (1), (2) and (3) are satisfied. The flow starts with core a (the upper left adjacent core). If the criteria are satisfied, core a is selected as the destined core for task migration; at the same time, the thermal and load parameters of the current core ($T_{eff,cur}, P_{cur}, TNP_{cur}$) are updated by the thermal and load parameters of core a ($T_{eff,a}, P_a, TNP_a$). Then the flow continues to check with core b, if the migration criteria are satisfied, it indicates that core b is a better migration choice with even lower $T_{eff,b}$, P_b, and TNP_b than the previous migration choice of the destined core, and thus will be selected to replace the previous choice as the destined core for task migration. The same procedure will be repeated for core c to core h. The flow continues to update the choice and will finally find the best choice for task migration among the 8 adjacent cores, which has the lowest value of temperature (T), load (P), and total load of all its adjacent cores (TNP). An example flow is shown in Fig. 5 (b) in which the circle nodes represent the cores, the filled grey nodes indicate the checked cores, and the filled black node indicates the chosen core for task migration at the end of the task migration method flow. In this example, the new method checks with all of the 8 adjacent cores (stamped with grey) and finally selected core c as the destined core (stamped with black) to migrate the task to.

In this way, the distributed thermal management method takes account of the task load, the influence of the task load of the adjacent cores, and the thermal transient effect due to the package properties to perform task migrations and optimize the thermal profile across all the cores.

D. The distributed average temperature tracking algorithm for the mode switching control logic

The mode switching control logic is part of the proposed thermal management method, and it will switch the fully utilized chip to low power mode if the average temperature is above a certain threshold. However, as discussed before, the average temperature cannot be calculated by taking the mean of the temperature values across all the microprocessor cores because each core is only aware of its own temperature and the temperature of its adjacent cores. Thus, a distributed state based average temperature tracking method is applied to estimate the average temperature across all the cores without monitoring the global temperature of the chip. Before



(a) task migration method flow (Applicable to any arbitrary core)

(b) A graphic example (the proper core is chosen)

Fig. 5. The proposed distributed task migration method

introducing the proposed estimation method, it is necessary to investigate some mathematic theories in distributed control world.

Distributed average tracking theory for multi-agent system has proved that the states of all the distributed agents could converge to the average value of the time-varying reference signals [14], [15]. According to [15], the state equation of agent i of the distributed system can be formulated as

$$\begin{aligned} \dot{z}_i(t) &= \alpha \sum_{j \in N_i} sgn[x_j(t) - x_i(t)] \\ x_i(t) &= z_i(t) + r_i(t) \end{aligned} \tag{10}$$

in which $x_i(t)$ is the local state of the distributed agent, $r_i(t)$ is the reference signal with bounded derivatives in a finite time, sgn is the sign function, and $\alpha > 0$ is a constant. N_i denotes the state at the neighborhood of agent i. The distributed system assumes that the value of each reference $r_i(t)$ is local to agent i only, and the value of the state $x_i(t)$ is known only by the distributed agents in its neighborhood.

Suppose that the changing rate of $r_i(t)$ is bounded by β. For the algorithm (10), our theoretical analysis in [16] shows that if the graph G is undirected and connected, and $\alpha > \beta$, then each estimate $x_i(t)$ approaches the average reference signal $\frac{1}{N}\sum_{j=1}^{N} r_j(t)$ in finite time, and the convergence time is guaranteed to be upper bounded by $\frac{1}{2(\alpha-\beta)}\sum_{i=1}^{N}\sum_{j\in\mathcal{N}_i}|r_i(0) - r_j(0)|$. Note that we can easily select the parameter α to ensure an arbitrary desired convergence time.

Due to the discrete-time nature of the update in multi/many core systems, in this project, we will apply a discrete-time version of the algorithm (10). Let k denote the discrete-time index, T_s denote the sampling period, and $r_i = f_i$ (recall that f_i denotes the temperature of agent or core i). The discrete-time version of the algorithm (10) is given as

$$\begin{aligned} z_i[k+1] &= z_i[k] + T_s\alpha \sum_{j\in\mathcal{N}_i} \text{sgn}(x_j[k] - x_i[k]) \\ x_i[k] &= z_i[k] + f_i[k], \quad i = 1, \ldots, N. \end{aligned} \tag{11}$$

In this way, the average temperature could be estimated through tracking the distributed state variable $x_i[k]$.

IV. Experimental results

A. Experiment setups

The proposed thermal management method is implemented using Matlab 7.0, and Hotspot [1] is used to build the thermal model based on the configuration of the many-core microprocessor and simulate the temperature responses. A 100-core processor system with 10×10 configuration as shown in Fig. 6 is used to evaluate the proposed method. Each core of the processor has geometry of 4 mm $\times$ 4 mm, and the thickness of the processor chip is 0.15 mm. More detailed thermal package structure and material properties are summarized in Table I, in which k denotes thermal conductivity, and c denotes specific heat. The heat convection coefficient of the top surface of the heat sink is set to 3000 J/K to model the convective cooling effect from the top. The sampling interval of the thermal data of Hotspot is set to be 30 μs to preserve simulation accuracy.

TABLE I

The package structure and thermal properties of the 100-core microprocessor.

Components	Chip	Heat Spreader	Heat Sink
Thickness(mm)	0.15	1.00	6.90
k ($W/(mK)$)	100.0	400.0	400.0
c ($J/(m^3K)$)	1.75×10^6	3.55×10^6	3.55×10^6

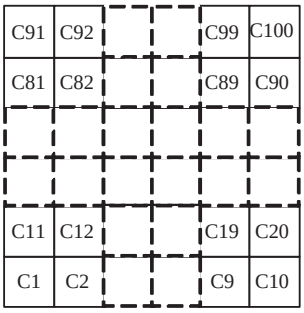


Fig. 6. The configuration of the 100-core microprocessor die (from core 1 (C1) to core 100 (C100)).

We used Wattch as the architecture level power analysis tool with modifications to extend its functionality to calculate power under different supply voltage and working frequency [17], and simulate the detailed transient power traces using 16 dynamic workloads (ammp, apsi, bzip2, equake, galgel, gcc, lucas, mesa, parser, twolf, vpr, applu, art, crafty, fma3d, gap) from SPEC2000 benchmarks. During the simulation, the proposed distributed task migration policy will be applied to each core to optimize the task assignment at the beginning of

each execution cycle (1800 time steps). Meantime, distributed tracking technique is used to estimate the average temperature distributively. If the estimated average temperature is below the threshold ($T_{th} = 75^oC$), the processor chip works under normal mode with full working frequency; otherwise, the mode switching control logic will switch the processor to low power mode with reduced working frequency and supply voltage (both working frequency and supply voltage are reduced to 90% of the original value under normal mode).

To evaluate the result of the proposed thermal management method quantitatively, we look at the variances of on-chip temperature: in the normal working mode, the proposed method should be able to lead to smaller temperature variance across the cores, and more balanced on-chip temperature distribution. We also look at the occurrence of thermal hot spots during the task execution: in the normal working mode, the proposed method should be able to more effectively reduce the number of thermal hot spots comparing to the existing and alternative methods. Finally, we check the mode switching control logic: the temperature tracking technique should give a good estimation of the average temperature, and the mode switching control logic could effectively prevent the overall temperature of the chip from increasing to an alarming level.

B. Performance evaluation of distributed thermal management

1) Performance comparison of the task migration schemes: This section compares the result of the proposed task migration schemes against the conventional ones and the recently reported DTB-M in [12]. Fig. 7 shows the comparison of transient temperature variance, in which the notation 'Dist-Teff' represents the proposed method; 'centralized-Teff' represents the centralized method using the proposed thermal indicator T_{eff}; 'centralized' represents the centralized method using real temperature; 'Dist' represents the distributed method using real temperature; 'DTB-M' is the existing method in [12]. As expected, the centralized methods achieves the best results in term of reducing the on-chip temperature variances. Among the two centralized control methods, the one uses T_{eff} achieves even smaller temperature variance comparing with the one using real temperature. Among the three distributed thermal management method, the proposed one that uses T_{eff} for task migration lead to less temperature variance than the counterpart that uses real temperature, and both are more efficient than the existing DTM-B in terms of temperature variance reduction.

Hence, this result clearly demonstrated that the effective initial temperature T_{eff} is a better thermal indicator comparing with real temperature, and the proposed distributed task migration method would lead to reduced temperature variance, yielding more balanced on-chip temperature distribution across all the cores comparing with other distributed methods.

Fig. 8 shows the statistic of thermal hot spot occurrence above different temperature levels during the task execution. Comparing with the centralized method using T_{eff}, the proposed distributed method shows comparable reduction of thermal hot spot where temperature is above 80 ^oC, although the centralized counterpart is obviously more efficient to reduce thermal hot spot above higher temperature. Also, the proposed

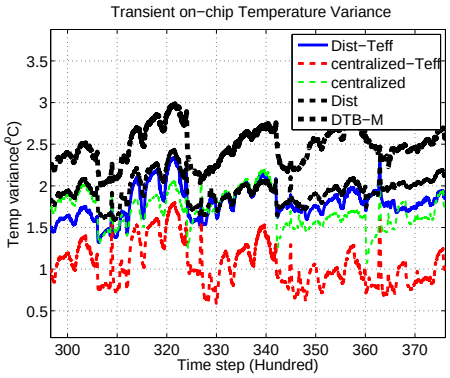


Fig. 7. Comparison of the transient temperature variance under different thermal management policy

distributed method shows even better results comparing with the centralized scheduling method that uses real temperature for task migration decision. Comparing with the distributed counterpart method that uses real temperature, the proposed method lead to 33% reduction of thermal hot spot occurrence during task executions. Comparing with the existing DTB-M, the number of thermal hot spots encountered during task executions is reduced by 42% under the proposed distributed task migration method. Thus, the proposed distributed task migration method can more effectively remove on-chip thermal hot spots and lead to more uniform temperature profile of many/multi-core microprocessors.

Besides, as expected, being able to globally optimize the task assignment, the centralized control method using T_{eff} shows the most efficient thermal hot spot reduction. In addition, the figure shows that the number of thermal hot spots over high temperature (temperature above 80 ^oC) is reduced quickly, which indicates that all these task migration methods could effectively reduce the temperature of over-heated core and thus prevent thermal emergency.

Also, the proposed method could effectively lower the overall temperature of the chip due to the optimized task assignment based on T_{eff}. Fig. 9 shows the comparison of average temperature of the chip employing different thermal management methods. Lowering the overall temperature also greatly contributes to the removal of thermal hot spot, which explains the reason why the proposed method is efficient in terms of thermal hot spot reduction.

2) Distributed tracking of global average temperature under the proposed mode switching control logic: Since the average temperature is used as an global thermal indicator to enable the mode switching control logic, an accurate average temperature estimation based on distributed states is the prerequisite for the mode switching control logic. Fig. 10 shows the result of the distributed average temperature estimation, which indicates that the distributed state of each core converges to the value of the average temperature of the whole many-core chip, and thus we could use the value of the state variable of each local core to represent the real-time on-chip global average temperature. In this way, the average temperature could be tracked distributively in each local core, and the mode switching control logic would switch the processor to lower power mode with reduced working

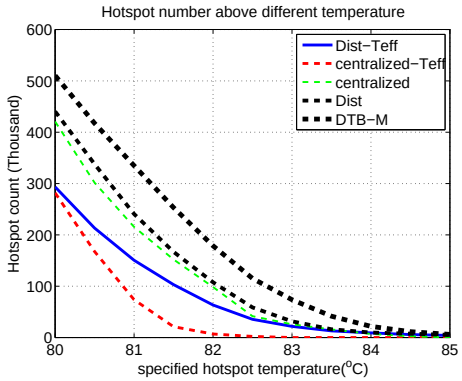


Fig. 8. The occurrence of thermal hot spots above different temperature levels during task execution.

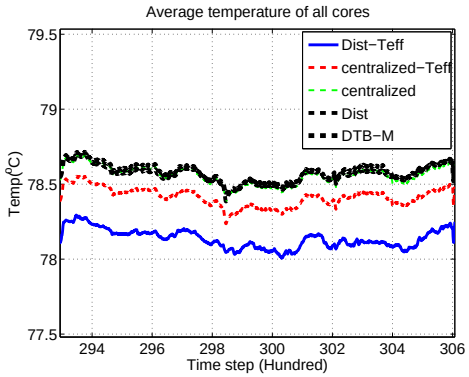


Fig. 9. Comparison of on-chip average temperature during task execution

frequency and supply voltage (90% of their original value in normal mode) if the estimated average temperature exceeds a certain threshold. Fig. 10 also shows how the average temperature is dynamically controlled by the mode switching control logic.

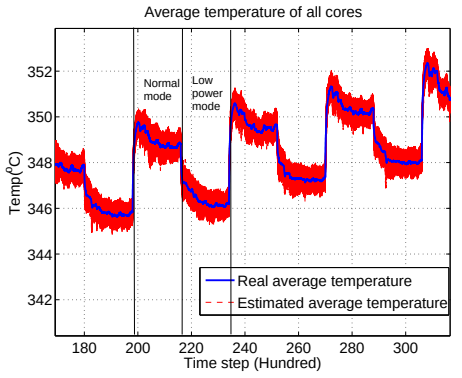


Fig. 10. The average temperature tracking based on distributed state tracking method under the proposed mode switching control logic

V. Conclusion

In this paper, a new distributed thermal management that uses task migration and mode switching control logic is proposed to reduce the on-chip temperature variance and thermal hot spots during the task execution. The new method is fully distributed in the sense that the task migrations happen only between two adjacent cores, and the average temperature estimation for mode switching control is performed locally in each individual core. By considering the influence of the load of adjacent cores and transient thermal effect using a new term called effective initial temperature, the proposed method lead to more balanced thermal profile. Our experimental results on a 100-core microprocessor shows that the proposed method works more effectively in reducing the on-chip temperature variance and thermal hot spots comparing with the existing distributed method (42% more thermal hot spot reduction) and the alternative method that does not consider the transient thermal effect (33% more thermal hot spot reduction). Also, it shows that the average temperature could be accurately estimated in a distributed way, which is used by mode switching control logic as an overall thermal indicator to switch the processor between normal mode and low power mode.

References

[1] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *Proc. Int. Symp. on Computer Architecture (ISCA)*, pp. 2–13, 2003.

[2] M. Hertl, D. Weidmann, and A. Ngai, "An advanced Reliability Improvement and Failure Analysis Approach to Thermal Stress Issues in IC Packages," in *35th International Symposium for Testing and Failure Analysis, ISTFA*, pp. 28–32, 2009.

[3] G. Liu, M. Fan, and G. Quan, "Neigbor-Aware Dynamic Thermal Management for Multi-core Platform," in *Proc. European Design and Test Conf. (DATE)*, pp. 187–192, 2012.

[4] A. Chakraborty, K. Duraisami, A. Sathanur, P. Sithambaram, A. Macii, E. Macii, and M. Poncino, "Implementation of a thermal management unit for canceling temperature-dependent clock skew variations," *Integration, the VLSI Journal*, vol. 41, no. 1, pp. 2–8, 2008.

[5] "International technology roadmap for semiconductors (ITRS), 2011edition," 2011. `http://public.itrs.net`.

[6] M. Powell, M. Gomaa, and T. N. Vijaykumar, "Heat-and-run: leveraging smt and cmp to manage power density through the operating systems," in *ACM SIGPLAN NOTICES*, vol. 39, pp. 260–270, 2004.

[7] I. Yeo, C. C. Liu, and E. J. Kim, "Predictive dynamic thermal management for multicore systems," in *Proc. Design Automation Conf. (DAC)*, DAC '08, (New York, NY, USA), pp. 734–739, ACM, 2008.

[8] A. K. Coskun, T. S. Rosing, and K. C. Gross, "Utilizing predictors for efficient thermal management in multiprocessor SoCs," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, pp. 1503–1516, October 2009.

[9] S. Borkar, "Thousand core chips: a technology perspective," in *Proc. Design Automation Conf. (DAC)*, pp. 746–749, 2007.

[10] "Intel's 48-core Single-Chip Cloud Computer." `http://www.intel.com/content/www/us/en/research/intel-labs-single-chip-cloud-computer.html`.

[11] T. Ebi, M. A. Al Faruque, and J. Henkel, "TAPE: thermal-aware agent-based power economy for multi/many-core architectures," in *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, ICCAD '09, (New York, NY, USA), pp. 302–309, ACM, 2009.

[12] Y. Ge, P. Malani, and Q. Qiu, "Distributed task migration for thermal management in many-core systems," in *Proc. Design Automation Conf. (DAC)*, pp. 579–584, 2010.

[13] L. T. Pillage, R. A. Rohrer, and C. Visweswariah, *Electronic Circuit and System Simulation Methods*. New York: McGraw-Hill, 1994.

[14] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, pp. 1520–1533, 2004.

[15] F. Chen, Y. Cao, and W. Ren, "Distributed average tracking of multiple time-varying reference signals with bounded derivatives," *IEEE Transactions on Automatic Control*, vol. PP, p. 1, 2012.

[16] F. Chen, Y. Cao, and W. Ren, "Distributed computation of the average of multiple time-varying reference signals," in *Proceedings of the American Control Conference*, (San Francisco, CA), pp. 1650–1655, July 2011.

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