Throughput Optimal Task Allocation under Thermal Constraints for Multi-core Processors [*]

Vinay Hanumaiah[‡], Ravishankar Rao[§], Sarma Vrudhula[‡], and Karam S. Chatha[‡]
[‡]Computer Science and Engineering Department, Arizona State University, Tempe, AZ 85287, USA
[§]Synopsys, Inc., Mountain View, CA 94043, USA
[‡]{vinayh, vrudhula, kchatha}@asu.edu
[§]ravirao@asu.edu

ABSTRACT

It is known that temperature gradients and thermal hotspots affect the reliability of microprocessors. Temperature is also an important constraint when maximizing the performance of processors. Although DVFS and DFS can be used to extract higher performance from temperature and power constrained single core processors, the full potential of multi-core performance cannot be exploited without the use of thread migration or task-to-core allocation schemes. In this paper, we formulate the problem of throughput-optimal task allocation on thermally constrained multi-core processors, and present a novel solution that includes optimal speed throttling. We show that the algorithms are implementable in real time and can be implemented in operating system's dynamic scheduling policy. The method presented here can result in a significant improvement in throughput over existing methods (5X over a naive scheme).

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems—*modeling techniques, performance attributes*; B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids

General Terms

Algorithms, Performance, Design, Theory

Keywords

Thermal management, multi-core processors, task allocation, optimal throughput, thread migration

1. INTRODUCTION

*This work was supported in part by NSF grant CSR-EHS 0509540, Consortium for Embedded Systems grant DWS 0086, and by a grant from Science Foundation Arizona (SFAz) and Stardust Foundation.

While the processor industry is aggressively scaling the number of cores on processors [3], power and thermal constraints pose a significant challenge to future multi-core processors. Currently, there are up to 64 cores available commercially on a single die [6]. The trend towards multi-core strategy is motivated by the fact that multi-cores can achieve higher performance at reduced core speeds by exploiting thread-level parallelism within a given power budget. However there are many challenges that need to be addressed in order to have hundreds or even thousands of cores in the near future [3]. The die temperature has to be maintained below the specified maximum temperature for safe and reliable operation of the processor. Cost and volume constrains the packaging and cooling solutions, and this necessitates the need to incorporate the Dynamic Thermal Management (DTM) techniques.

The traditional focus of performance optimization has been to maximize performance subject to constraints on power or energy consumption. This is a different and easier problem than optimizing performance under thermal constraints because the power budget is a single upper bound on the total power consumed by all units in all cores, whereas the thermal constraint is an upper bound that must be satisfied by each and every unit. Moreover, the relation between power and speed is a much simpler one (simple algebraic) than the temperature-speed relation (requires solution to hundreds of equations). Thus the traditional power management techniques such as Dynamic Frequency Scaling (DFS) and Dynamic Voltage and Frequency Scaling (DVFS) must also account for temperature.

1.1 Related Work

While there are DTM methods to guarantee optimal performance in case of single core processors, the same is not true in the case of multi-core processors, because of the larger dimension of the problem. Local optimization for a single core is not necessarily optimal for a multi-core processor [20]. Apart from speed and voltage control, task migration or task-to-core allocation provides an additional dimension for performance optimization. Although a significant body of research has recently started on the problems of performance optimization of multi-cores under thermal constraints [5, 8, 10, 13], there are no optimal methods that incorporate task-to-core allocation.

A detailed comparison of various techniques aimed at general purpose, single core processors appears in [4]. The problem of optimal speed control for a sequence of tasks on a single core was addressed in [19]. DVFS with thermal constraints for a periodic sequence of tasks on a single core was addressed in [22]. In [12], the problem of *off-line* multi-core speed control under thermal and power constraints is solved through convex optimization. In [12, 22] the leakage dependence on temperature, which substantially complicates the temperature-power relation, is not considered. In [20], the authors provided an online computational model to calculate the speeds of cores, for both transient and steady-state cases. Leakage dependence on temperature (LDT) is considered in [20] and an accurate thermal model (HotSpot [14]) is included. A comprehensive summary of thread migration techniques is provided in [11, 13]. The authors of [8] proposed a scheme called *heat-and-run* which moves threads from over heated SMT cores to cooler cores for maximizing performance. While this technique works when the number of tasks is less than number of cores and for processors that have a temperature slack, it may not be optimal for high performance processors where most of the cores operate close to thermal maximum. In [5] various thermal management techniques are studied, including OS based migration controllers. They make use of multi-loop control, wherein thread migration controls the outer loop, and DVFS makes up the inner loop.

Previous studies on task migration methods [8,11,13] used detailed cycle accurate simulators to determine the migration policies. The simulation times can be as large as tens of hours. These are not suitable for thread assignment in real time. Hence, there is a need to make use of accurate architectural thermal models with suitable approximations so that the run time of the algorithm is close to the migration interval.

1.2 Main Contributions and Outline

In this paper we present a complete formulation of the throughput-optimal task-to-core allocation problem under thermal constraints. This turns out to be a computationally expensive non-linear optimization problem, that cannot be solved in real time. We then present a novel simplification of the problem that allows us to compute the optimal task assignment and throughput-optimal speed at the start of each migration interval. This is then combined with a throughput-optimal continuous speed function over the migration interval. The problem formulation includes a detailed thermal model that accounts for thermal characteristics of the functional units within each core, the thermal interface material, the package and the intra-core and inter-core heat flow. We demonstrate, through simulation, that throughput improvements over a naive scheme can be as high as 5X. The proposed algorithm is feasible for on-line computation as it can be executed within a reasonable OS scheduling interval of $\approx 5ms$, for a large number of cores and tasks (as much as 128 cores and 128 tasks).

The paper is organized as follows: Section 2 introduces the system models and notations. It explains the HotSpot model [14] and the piece-wise linear (PWL) model for decoupling LDT. The problem of transient throughput optimization is introduced in Section 3. We also explain the structure of the problem and assumptions made in order to reduce the problem complexity. Section 4 contains a polynomial time algorithm for maximizing the instantaneous throughput. Finally Sections 5 and 6 present results and conclusion respectively.

2. SYSTEM MODELS AND NOTATIONS

2.1 Performance Model

We consider a heterogeneous multi-core processor with a queue containing n_t non-identical tasks to be run on n cores. We assume that each core i can be assigned an independent speed s_i, which is normalized and can be continuously varied over [0, 1]. We do not use dynamic voltage scaling in this work for the following reasons: (i) speed control through clock gating or fetch throttling can be activated with less overhead, and (ii) supply voltages are already small and are not expected to scale any further in future generations, and this leaves a very small margins for scaling [21].

In our work we do not assume SMT (simultaneous multi-threading) for the reasons of simplicity and also due to the fact that simple single-context cores are expected to be more power efficient [3]. We define the throughput of a multi-core processor as the weighted sum of the speeds of all cores, i.e. $S = \int_0^t \Sigma_{i=1}^n w_i s_i(t) dt$. The weights could be the IPC (instructions per clock cycle), the task priorities or any other performance criterion that varies with tasks. For inactive cores (those cores without tasks assigned), the weights $w_i = 0$. When IPCs are used as weights, the throughput reduces to instructions per second.

We assume that the threads run for a duration of at least the die thermal constant (around few milliseconds). Class A and Class B of NAS benchmarks [1] fall into this category. This allows the thermal die capacitances to saturate and hence they can be ignored without any significant loss of accuracy.

2.2 Power and Thermal Models

We use the latest HotSpot thermal RC circuit model [14] to model the thermal behavior of a processor. HotSpot uses the duality between heat flow and electrical circuit phenomena. Heat transfer and retaining capacity are represented by resistances and capacitances respectively, while the heat generating sources are modeled through current sources. Figure 1 taken from [18] shows HotSpot thermal model for a four core processor. Each core on the die is divided into several blocks (4 shown), with the same division being applied to the Thermal Interface Material (TIM). The heat spreader and the heat sink are divided into a number of blocks as shown. For a processor with n cores and m functional units for each core, there will be nm blocks in the die, another nm blocks in the TIM and 14 blocks in the package according to HotSpot-4 model [14], for a total of $N = 2nm + 14$ blocks.

Execution of various tasks on cores creates a spatial and temporal distribution of temperature on the die. Spatial variations arise due to different circuit styles, their size, different core speeds, activity factor of tasks, etc., whereas, temporal variations arise due to the time-varying nature of the code and core speeds, context switching among the threads, etc. The system power and temperature vectors[1] are represented by $\mathbf{P}$ and $\mathbf{T}$ respectively. The power $\mathbf{P}$ is the sum of the dynamic power component $\mathbf{P}_d$ and the leakage power component $\mathbf{P}_s$. The dynamic power depends only on speed $\mathbf{s}(t)$, whereas the leakage power depends only on temperature $\mathbf{T}(t)$. Hence $\mathbf{P} = f(\mathbf{s}(t), \mathbf{T}(t))$.

The relationship between the temperature vector $\mathbf{T}$ and

[1]All vectors are assumed to be column vectors and variables in bold represent matrices.

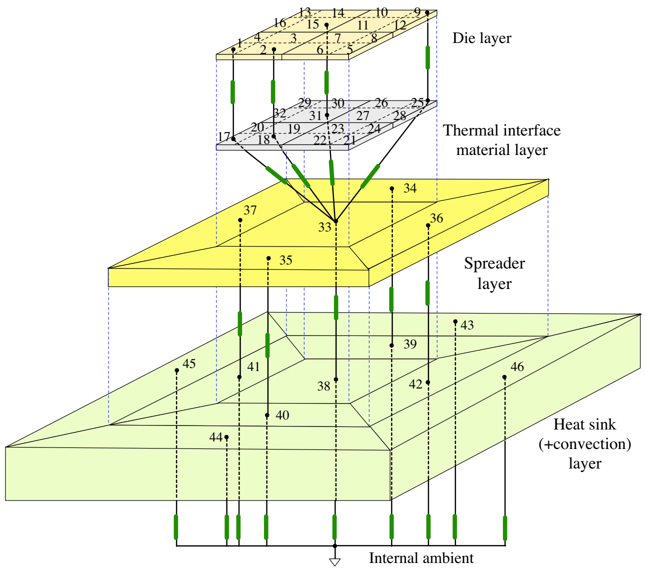
Die layer
Thermal interface material layer
Spreader layer
Heat sink (+convection) layer
Internal ambient

Figure 1: HotSpot thermal model for a four core processor [18].

the power vector $\mathbf{P}$ is given by the following dynamic state-space differential equation [7,16]:

$$\frac{d\mathbf{T}}{dt} = \mathbf{AT} + \mathbf{BP}(\mathbf{s}, \mathbf{T}, t) \tag{1}$$

Given a floorplan of a multi-core processor, the $N \times N$ ($N = 2nm + 14$) matrices, $\mathbf{A}$ and $\mathbf{B}$ are determined from HotSpot [7,14] model. The vectors $\mathbf{P}$ and $\mathbf{T}$ are of length N. However $\mathbf{P}$ only has nm non-zero components because the remaining blocks of the TIM and the package do not generate heat.

In order to decouple the cyclic dependency between power and temperature, we need to linearize the non-linear relation between leakage and temperature. From [20], we note that the leakage power can be represented as $\mathbf{P}_s = \mathbf{P}_{s0} + \mathbf{G}_{\mathrm{LDT}}\mathbf{T}$, where $\mathbf{G}_{\mathrm{LDT}}$ is a matrix whose elements represent the slopes of the power-temperature curve. Let us define $\mathbf{P}_d^{max}$ as the vector of maximum power dissipation when $\mathbf{s} = \mathbf{1}_{n\times 1}$ (constant vector of 1's). The dynamic power is given by $\mathbf{P}_d(\mathbf{s}) = diag(\mathbf{P}_d^{max})\mathbf{Xs}$, where $\mathbf{X}$ is given by,

$$X(i,j) = \begin{cases} 1, & \text{if block } i \text{ belongs to core } j \text{ and } i \leq nm, \\ 0, & \text{otherwise,} \end{cases}$$

The state space equation (1) can now be re-written in standard state space form as follows:

$$\begin{aligned} \frac{d\mathbf{T}}{dt} &= \mathbf{AT} + \mathbf{B}(\mathbf{P}_d(\mathbf{s}) + \mathbf{P}_{s0} + \mathbf{G}_{\mathrm{LDT}}\mathbf{T}) \\ &= \hat{\mathbf{A}}\mathbf{T} + \mathbf{B}(\mathbf{P}_d(\mathbf{s}) + \mathbf{P}_{s0}) \end{aligned} \tag{2}$$

where $\hat{\mathbf{A}} \triangleq \mathbf{A} + \mathbf{BG}_{\mathrm{LDT}}$.

3. PROBLEM DEFINITION

Given a heterogeneous multi-core processor with n cores, a weight vector $\mathbf{w}$, power vectors $\mathbf{P}_d^{max}$ (of all tasks) and $\mathbf{P}_{s0}$, conductance matrix $\mathbf{G}$ and spreader temperature T_{spr}, find an optimal allocation of n_t (non-identical) tasks on n cores with optimal speeds of operation such that the total throughput defined by $\frac{1}{t_s}\int_0^{t_s} \mathbf{w}^T\mathbf{s}(t)dt$ is maximized over the migration interval [t_0, t_s], subject to the constraints that the temperature of no functional block is greater than the allowed maximum temperature T_{max} and frequency ranges over [0, 1].

$$\max_{\mathbf{s}(t),\mathbf{M}} \quad \frac{1}{t_s}\int_0^{t_s} \mathbf{w}^T\mathbf{s}(t)\, dt, \tag{3}$$

$$s.t. \quad \frac{d\mathbf{T}}{dt} = \hat{\mathbf{A}}\mathbf{T} + \mathbf{B}(\mathbf{P}_{dt}^{max}\mathbf{M}^T\mathbf{s}(t) + \mathbf{P}_{s0}), \tag{4}$$

$$\sum_{i=1}^{n} \mathbf{M}(i,j) = 1, j \in \{1,\ldots,n_t\}, \tag{5}$$

$$\sum_{j=1}^{n_t} \mathbf{M}(i,j) = 1, i \in \{1,\ldots,n\}, \tag{6}$$

$$\mathbf{M}(i,j) \in \{0,1\}, i \in \{1,\ldots,n\}, j \in \{1,\ldots,n_t\}, \tag{7}$$

$$\mathbf{T}(0) = \mathbf{T_0}, \tag{8}$$

$$\mathbf{T}(t) \leq \mathbf{T}_{max},\ \forall\ 0 \leq t \leq t_s, \tag{9}$$

$$\mathbf{0}_{n\times 1} \leq \mathbf{s}(t) \leq \mathbf{1}_{n\times 1},\ \forall\ 0 \leq t \leq t_s \tag{10}$$

where, $\mathbf{P}_{dt}^{max}$ is a matrix whose columns contain $\mathbf{P}_d^{max}$ of n_t tasks and is of dimension $N \times n_t$. $\mathbf{T}_0$ is the vector of initial temperatures. $\mathbf{M}$ is the mapping/allocation matrix of size $n \times n_t$. The elements of $\mathbf{M}$ are defined as follows:

$$M(i,j) = \begin{cases} 1, & \text{if task } j \text{ is assigned to core } i, \\ 0, & \text{otherwise,} \end{cases}$$

Equation (4) represents a time-dependent non-linear constraint involving variables $\mathbf{s}(t)$ and $\mathbf{M}$. Performing non-linear optimization in real time is computationally not practical. This motivates us to simplify the problem as follows.

Problem Simplification: Owing to penalty of high migration overhead, the task allocation is done once in tens of milliseconds. A significant simplification is obtained if we restrict the problem to finding an allocation that maximizes the *instantaneous* throughput at the start of the migration interval. Then with those corresponding initial speeds, we can find the throughput-optimal speed curve over the migration interval. Letting $\mathbf{s}_0$ denote the vector of initial speeds at the start of a migration interval, the objective will be to maximize the weighted sum of initial speeds, $\mathbf{w}^T\mathbf{s}_0$. In order to solve for the initial speeds $\mathbf{s}_0$, we need to decouple the dependence of $\mathbf{M}$ on $\mathbf{s}_0$ and vice-versa. In the following sections we show how to decouple the above dependency by making a few simple, but realistic assumptions.

4. OPTIMAL PLACEMENT ALGORITHM

4.1 Structure of HotSpot Conductance Matrix

We build multi-core floorplan from single core floorplans with L2 caches surrounding these cores. Intel Core 2 Duo processor has a similar floorplan. Given the geometric floorplan, one can calculate the conductance matrix $\mathbf{G}$ using HotSpot thermal circuit model. Figure 2 taken from [18] shows the sparsity plot of the conductance matrix for the dual core Alpha processor constructed as explained above. A dot in the plot corresponds to a non zero entry in the matrix. We can see in the figure that most of the entries are concentrated along the diagonal blocks of the core and the TIM. The size of the blocks correspond to the number of functional blocks m. These square diagonal blocks represent the lateral resistances of the core and TIM. The sets of dots running in parallel to the main diagonal represent the vertical resistances connecting the die and the TIM layers. The vertical resistances between the TIM and the spreader are shown by the vertical and the horizontal strip of dots in the figure.

Caches separate the cores and they tend to be the coolest parts of the die as they have larger die area and the resulting low power density and temperature [9]. This results in smaller lateral resistances between cores as seen in Figure 2.

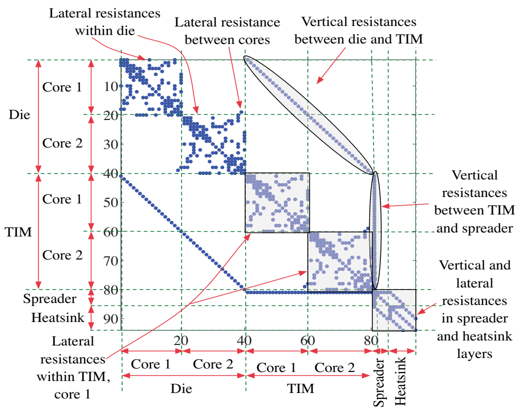

Figure 2: Plot of conductance matrix sparsity for the constructed dual core Alpha floor plan [18].

Figure 3 taken from [18] shows the various components of the conductance matrix shown in Figure 2. The diagonal components of the matrix are named $\mathbf{G}_{\text{die}}$ and $\mathbf{G}_{\text{tim}}$ corresponding to the die and the TIM sections of the chip. The vertical conductances connecting the die to the TIM are denoted by $\mathbf{G}_{\text{die-tim}}$ and $\mathbf{G}_{\text{tim-die}}$, which are identical as it is the same conductance that connects both the die and the TIM layer. $\mathbf{G}_{\text{tim-spr}}$ and $\mathbf{G}_{\text{spr-tim}}$ denote the conductances connecting the spreader and the TIM layer. Finally the conductances in the package are denoted by $\mathbf{G}_{\text{pkg}}$.

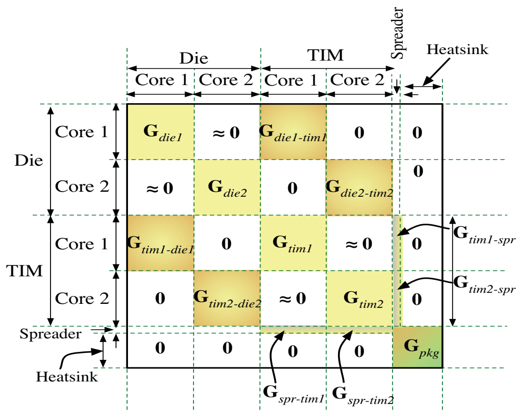

Figure 3: Components of the thermal conductance matrix G for the constructed Alpha dual core [18].

We make following observations based on the conductance matrix shown in Figure 3:

- There are fewer lateral resistances between cores and since the cores are surrounded by caches, the heat flow between the cores can be neglected (as caches are the cooler parts of the chip).
- The lateral thermal resistances are nearly four times higher than the vertical resistances in the die and the TIM layers [19]. Thus majority of the heat flow occurs through the vertical resistances. Also since the vertical thermal gradient is larger between the spreader and the TIM than across the die, we can ignore the lateral resistances.
- The TIM has lower thermal conductance than the silicon, which further reduces the heat flow in that layer.

Based on the above observations, we neglect the sparse blocks of the matrix corresponding to inter-core die and TIM resistances.

4.2 Computation of Die Temperatures

The total power dissipation in the processor is given by:

$$\mathbf{GT} = \mathbf{P}_d(\mathbf{s}) + \mathbf{G}_{LDT}\mathbf{T} + \mathbf{P}_{s0} \tag{11}$$

Using Kirchhoff's current law for the die and TIM layers, the following equation can be derived for core i running task j:

$$\mathbf{G}_{\text{die,i}}\mathbf{T}_{\text{die,i}} + \mathbf{G}_{\text{die-tim,i}}\mathbf{T}_{\text{tim,i}} = s_{ij}\mathbf{P}_{d,j}^{max} + \mathbf{G}_{\text{LDT,i}}\mathbf{T}_{\text{die,i}} + \mathbf{P}_{s0,i} \tag{12}$$

$$\mathbf{G}_{\text{tim-die,i}}\mathbf{T}_{\text{die,i}} + \mathbf{G}_{\text{tim,i}}\mathbf{T}_{\text{tim,i}} + \mathbf{G}_{\text{tim-spr,i}}T_{\text{spr}} = 0. \tag{13}$$

T_{spr} is the scalar temperature of the spreader center. Given $\mathbf{P}_{d_i}^{max}$ for task j and T_{spr}, we can find the maximum operational frequency s_{ij} of any core i. Let,

$$\mathbf{K}_{g,i} = \mathbf{G}_{\text{die,i}} - \mathbf{G}_{\text{die-tim,i}}\mathbf{G}_{\text{tim,i}}^{-1}\mathbf{G}_{\text{die-tim,i}} - \mathbf{G}_{\text{LDT,i}} \tag{14}$$

$$\mathbf{K}_{p,i} = \mathbf{G}_{\text{die-tim,i}}\mathbf{G}_{\text{tim,i}}^{-1}\mathbf{G}_{\text{tim-spr,i}}T_{\text{spr}} + \mathbf{P}_{s0,i}. \tag{15}$$

Let $\mathbf{T}_{\text{die,ij}}$ be the die temperature of core i when executing task j at the maximum speed and is given by:

$$\mathbf{T}_{\text{die,ij}} = \mathbf{K}_{g,i}^{-1}\mathbf{K}_{p,i} + \mathbf{K}_{g,i}s_{ij}\mathbf{P}_{d,j}^{max}. \tag{16}$$

Note that we need not be concerned with T_{tim}, since the hottest blocks will be part of the die. Now the problem stated in Section 3 can be re-stated as follows:

$$\max_{\mathbf{s},\mathbf{M}} \quad \mathbf{w}^T\mathbf{s}, \tag{17}$$

$$s.t. \quad \mathbf{T}_{\text{die,i}} = \mathbf{K}_{g,i}^{-1}\mathbf{K}_{p,i} + \mathbf{K}_{g,i}s_i\mathbf{P}_{dt}^{max}\mathbf{M}(i)^T, \quad \forall i \in \{1,\ldots,n\} \tag{18}$$

$$\sum_{i=1}^{n}\mathbf{M}(i,j) = 1, j \in \{1,\ldots,n_t\}, \tag{19}$$

$$\sum_{j=1}^{n_t}\mathbf{M}(i,j) = 1, i \in \{1,\ldots,n\}, \tag{20}$$

$$\mathbf{M}(i,j) \in \{0,1\}, i \in \{1,\ldots,n\}, j \in \{1,\ldots,n_t\}, \tag{21}$$

$$T_{spr} = T_{spr,0}, \tag{22}$$

$$\mathbf{T}_{\text{die,i}} \leq \mathbf{T}_{max}, \tag{23}$$

$$\mathbf{0}_{n\times 1} \leq \mathbf{s} \leq \mathbf{1}_{n\times 1} \tag{24}$$

where, $\mathbf{M}(i)$ refers to the i^{th} row of the matrix $\mathbf{M}$.

Equation (18) allows us to compute the speed s_i for a core i running task j independent of other tasks running on the remaining cores. The maximum of $\mathbf{T}_{\text{die},ij}$ should be less than the maximum allowed temperature T_{max}. Thus the hottest functional unit determines the maximum speed of operation for a given task j on core i. The hottest unit h is given by the row corresponding to the $row_h(\mathbf{T}_{\text{die},ij}) = max(\mathbf{T}_{\text{die},ij})$. The speed s_{ij} is given by:

$$s_{ij} = \begin{cases} \frac{T_{max} - row_h(\mathbf{K}_{g,i}^{-1}\mathbf{K}_{p,i})}{\mathbf{K}_{g,i}\mathbf{P}_{d,j}^{max}}, & max(\mathbf{T}_{\text{die,ij}}^{max}) \geq T_{max}, \\ 1, & \text{otherwise}, \end{cases} \tag{25}$$

This removal of dependency allows us to compute the speeds of a core for all available tasks and similarly for all other cores. Thus we can construct a matrix of speeds $\mathbf{S}$ whose elements are s_{ij}, where i, j refer to the core and task number respectively. We refere to this matrix as the *speed matrix*.

Given the above *speed matrix* we need to find an optimal task-to-core allocation. This problem can be formally stated as shown below:

$$\max_{\mathbf{M}} \quad \sum \mathbf{M}^T \mathbf{S}, \tag{26}$$

$$s.t. \quad \sum_{i=1}^{n} M(i,j) = 1, j \in \{1, \ldots, n_t\}, \tag{27}$$

$$\sum_{j=1}^{n_t} M(i,j) = 1, i \in \{1, \ldots, n\}, \tag{28}$$

$$M(i,j) \in \{0,1\}, i \in \{1, \ldots, n\}, j \in \{1, \ldots, n_t\}. \tag{29}$$

This is a linear assignment problem and can be efficiently solved in polynomial time using *Munkres* algorithm [17]. The time complexity of this algorithm is $O((nn_t)^3)$. Once the allocation is known along with the optimal initial speeds $\mathbf{s}_0$, the speed profile for the core i for the migration interval $[t_o, t_f]$ is given by [20]:

$$s_i(t) = s_{i0}e^{-t/\tau_i} + s_{i,ss}[1 - e^{-t/\tau_i}]. \tag{30}$$

where the τ_i is the time constant for the core i. The steady state speeds $s_{i,ss}$ and time constants τ_i are determined by solving the linear and non-linear program described in [20].

5. RESULTS

We used HotSpot thermal model to model the thermal behavior of the dual core Alpha 21264 processor constructed as described in Section 4.1. To construct the thermal model for the multi-core Alpha processor, we replicate the thermal model of one single core processor to desired number of cores by placing them side-by-side. The power numbers for the tasks were generated by uniformly distributing the power dissipation values obtained from SPEC benchmarks [2] using PTscalar tool [15]. We allowed a maximum temperature of 110 °C and 130 W of dynamic power and 60 W of leakage power. The maximum frequency of operation was set to 4 GHz. Spatial variation among cores is created by choosing a mean leakage power and varying it over a uniform distribution.

Chaparro et al. [10] have shown that performing thread assignment often can degrade performance. In fact they show that when the migration penalty is more than 2% of the total performance, the throughput starts degrading. The migration cost is assumed to be 40,000 cycles. For a 4GHz processor this approximates to 10 μs. In order to keep the migration penalty low and also since our tasks are assumed to run for at least the die thermal time constant, we set the migration interval to 10 ms or 40 million clock cycles . To the best of our knowledge there are no available literature on task-to-core allocation for thermally over constrained multi-core processors and hence chose to compare with round robin scheme.

5.1 Performance Comparison against Round Robin Scheme

Figure 4 shows the plot of the throughput of optimal allocation scheme versus round robin allocation for a four core processor with eight tasks. The simulation is run for a duration of 200 seconds with initial spreader temperature of 55 °C. The throughput between migrations is obtained by exponentially throttling the initial speed as described in [20]. We see a large improvement in throughput achieved by our optimal allocation algorithm. The throughput drops in both procedures after a few seconds as the package temperature increases necessitating the throttling of the cores and continues to drop until the cores attain the steady state speeds after package time constant ($\approx$ 100 s).

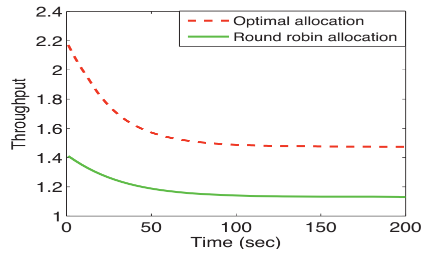

Figure 4: Temporal performance of optimal allocation and round robin allocation schemes.

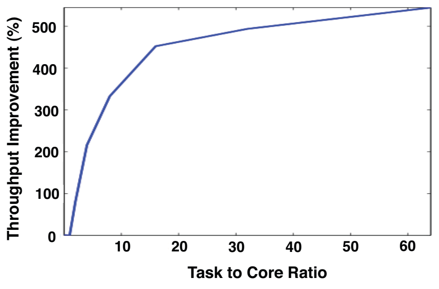

Figure 5: Plot of performance of the algorithm versus processor utility factor

5.2 Effect of Number of Cores and Tasks

Figure 5 shows the performance of the algorithm in optimally determining the allocation of tasks to cores against a random allocation. The improvement is described in terms of percentage throughput improvement and is plotted for several task to core ratios. We see that our algorithm shows a very high improvement when the processor utilization factor is very high. We see that the improvement can be as high as 5X.

5.3 Effect on Temperature Gradient

Figure 6 shows the spatial spatial distribution of temperature for a four core processor running optimal alloca-

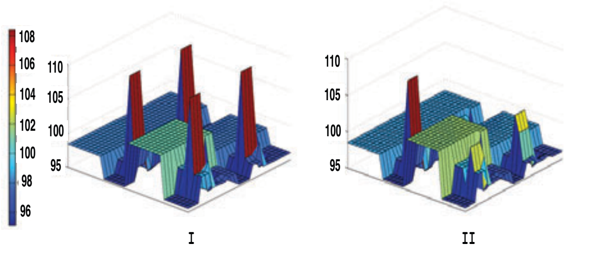

Figure 6: Spatial thermal map of 4-core processor utilizing I. optimal, II. random allocation algorithm

tion algorithm versus random allocation. To obtain maximum performance the total power dissipation has to be minimized and this fact is exploited by our procedure in seeking higher performance, yet minimize power dissipation and hence hotspots. The peaks in the plot represent the hotspots. This example explains the effectiveness of the optimal against random allocation algorithm in reducing the temperature deviation across the die.

5.4 Computation Time

Figure 7 shows the plot of the computation time for various size of the *speed matrix*. The algorithm is implemented in Matlab and is run on Intel core 2 duo at 2.16 GHz with 2GB RAM. The number of cells in the matrix is given by the product of number of cores and number of tasks. We see that the computation time is proportional to the number of cells. (**Note:** The x-axis is logarithmic, while the y-axis is linear). For the largest size of 16384 cells (128 cores and 128 tasks), the time needed to determine the optimal allocation is around 200 ms. Assuming that if there is a core dedicated to determine the allocation and the algorithm executed in binary format, we can fairly assume an 100X speedup over the Matlab implementation. This shows that even for a large number of cores and tasks it is possible to determine the allocation within the OS scheduling interval ($\approx$ 5 ms).

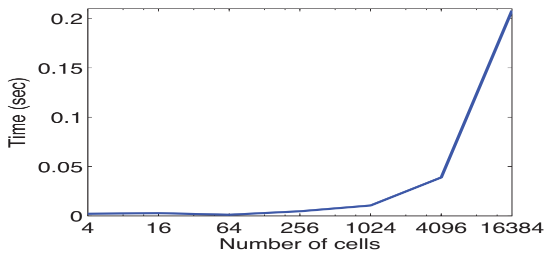

Figure 7: Plot of computation times as a function of number of cells in the *speed matrix*

6. CONCLUSION

Although multi-core processors can deliver higher performance, as they grow from tens to hundreds of cores, their power dissipation will also increase proportionately. With such large numbers of cores, it is unlikely that packaging alone will be sufficient to remove the heat, and dynamic thermal management will have to be employed far more frequently than with single core processors. In this work, we have used the frequency scaling and task-to-core allocation schemes of thermal management to optimally increase the performance of multi-core processors under thermal constraints. We formulated the combined problem as a linear assignment problem and showed that significant improvement in performance can be achieved over existing techniques (as high as 5X). The algorithm determines the optimal task-to-core allocation and subsequent speed control, accounting for the dependence of leakage on temperature and using an accurate thermal model (HotSpot). Finally, the proposed algorithm is sufficiently fast to be implementable in real time, within the OS.

7. REFERENCES

[1] NAS Parallel Benchmarks. http://www.nas.nasa.gov/Resources/Software/npb.html.

[2] SPEC CPU2000 Benchmarks. http://www.spec.org/benchmarks.html.

[3] S. Borkar. Thousand core chips: A technology perspective. In *DAC*, pages 746–749, 2007.

[4] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proc. HPCA*, pages 171–182, 2001.

[5] J. Donald and M. Martonosi. Techniques for multicore thermal management: Classification and new exploration. *SIGARCH Comput. Archit. News*, 34(2):78–88, 2006.

[6] D. Wentzlaff et al. On-chip interconnection architecture of the Tile Processor. *IEEE Micro*, 27(5):15–31, 2007.

[7] K. Skadron et al. Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management. In *Proc. HPCA'02*, pages 17–28, 2002.

[8] M. D. Powell et al. Heat-and-run: Leveraging SMT and CMP to manage power density through the operating system. *SIGOPS Oper. Syst. Rev.*, 38(5):260–270, 2004.

[9] M. Monchiero et al. Power/performance/thermal design-space exploration for multicore architectures. *IEEE Trans. Parallel Distrib. Syst.*, 19(5):666–681, 2008.

[10] P. Chaparro et al. Understanding the thermal implications of multicore architectures. *TPDS*, 18(8):1055–1065, 2007.

[11] P. Michaud et al. A study of thread migration in temperature-constrained multicores. *ACM Trans. Archit. Code Optim.*, 4(2):9–1–9–28, 2007.

[12] S. Murali et al. Temperature-aware processor frequency assignment for MPSoCs using convex optimization. In *Proc. CODES+ISSS*, pages 111–116, 2007.

[13] T. Constantinou et al. Performance implications of single thread migration on a chip multi-core. *ACM SIGARCH*, 33(4):80–91, 2005.

[14] W. Huang et al. An improved block-based thermal model in hotspot 4.0 with granularity considerations. In *WDDD*, 2007.

[15] W. Liao et al. Temperature and supply voltage aware performance and power modeling at microarchitecture level. *TCAD*, 24(7):1042–1053, 2005.

[16] Y. Han et al. Temptor: A lightweight runtime temperature monitoring tool using performance counters. In *TACS*, pages 17–28, 2006.

[17] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[18] R. Rao. *Fast and accurate techniques for early design space exploration and dynamic thermal management of multi-core processors*. PhD thesis, Arizona State University, 2008.

[19] R. Rao and S. Vrudhula. Performance optimal processor throttling under thermal constraints. In *Proc. CASES*, pages 257–266, 2007.

[20] R. Rao and S. Vrudhula. Efficient online computation of core speeds to maximize the throughput of thermally constrained multi-core processors. In *Proc. ICCAD*, pages 537–542, 2008.

[21] Y. Taur. CMOS design near the limit of scaling. *IBM J. Res. and Dev.*, 46(23):213–222, 2002.

[22] S. Zhang and K. S. Chatha. Approximation algorithm for the temperature-aware scheduling problem. In *Proc. ICCAD*, pages 281–288, 2007.