Efficient online computation of core speeds to maximize the throughput of thermally constrained multi-core processors

Ravishankar Rao and Sarma Vrudhula
Department of Computer Science and Engineering
Arizona State University, Tempe, Arizona 85281, USA
Email: {ravi.rao, vrudhula}@asu.edu

***Abstract*— We address the problem of efficient online computation of the speeds of different cores of a multi-core processor to maximize the throughput (which is expressed as a weighted sum of the speeds), subject to an upper bound on the core temperatures. We first compute the solution for steady-state thermal conditions by solving a linear program. We then present two approaches to computing the transient speed curves for each core: (i) a local solution, which involves solving a linear program every time step (of about 10 ms), and (ii) a global solution, which computes the optimal speed curve over a large time window (of about 100 s) by solving a non-linear program. We showed that the local solution is insensitive to the weights assigned in the performance objective (hence the need for the global solution). This is because a reduction in the speed of a core can only reduce the temperature of the other cores over much larger time periods (of the order of several seconds). The local solution is then completely determined by the temperature constraint equations. We show that the constraint matrix exhibits a special property - it can be expressed as the sum of a diagonal matrix and a matrix with identical rows. This allows us to solve the multi-core thermal constraint equations analytically to determine the (temporally) local optimum speeds. Further, we showed that due to this property, the steady-state speed solution selects a set of threads to operate at maximum temperature, and turns off all unused cores. Hence, to ensure that all available threads are scheduled, we impose a "fairness" constraint. Finally, we show how the open-loop speed control methods proposed above could be used together with a feedback controller to achieve robustness to model uncertainty.**

I. Introduction

The processor industry plans to aggressively scale the number of cores every two years. Already, up to eight cores are commercially available on a single die. This trend towards multi-core was motivated in part by power/thermal constraints on scaling clock frequencies. By reducing the clock frequency of each core, multi-core processors have been able to stay within the same power envelope, while achieving performance scaling by exploiting thread-level parallelism (TLP). However, there are several challenges that must be addressed to sustain this multi-core strategy [1]. One of them involves the growing importance of dynamic thermal management (DTM) strategies.

Consider an eight-core processor. During the course of operation, the number of threads deployed (and hence the number of active cores) will vary, which results in a wide range of die temperatures. This presents a new choice to system architects and package designers. They can design an expensive package that supports all cores running at full speed, or a cheaper package that uses DTM techniques to reduce the speeds when more than a critical number of cores (say 70% of them) are active. Allowing some throttling at higher loads enables the processor to operate at higher speeds at lower loads. Intel's Dynamic Accelaration Technology for dual-core processors already allows a core's speed to be increased if the other core is inactive, so that the available power budget can be fully utilized. The above argument points to the challenges and opportunities in extending this technology for many-core processors [2].

This work was funded by National Science Foundation grant CSR-EHS-0509540. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. An extended version of this paper is at http://veda.eas.asu.edu/bibadmin/show.php?id=174.

As DTM techniques become more common, it is important to extract maximum throughput under thermal constraints. In general, this requires solving large optimization problems at run-time. Such open-loop solutions must then operate in concert with multi-input multi-output feedback controllers to provide a robust optimal control strategy. Further, any overhead required to perform the above computation must be less than the performance improvement it provides. As the processor load can change during every operating system scheduling time slice (about 10 ms), it is important to determine the optimum speed combination with less than 1% overhead (i.e. in less than 0.1 ms). In this work, we present an analytical solution that can be computed very efficiently every 10 ms. We also propose a global solution that whose amortized overhead is very small.

A. Related work

A number of researchers have investigated distributed throttling schemes for multi-core processors [3]–[7] using cycle-accurate simulators. Most of them [3], [4] tried to maximize performance under a power budget. But this does not guarantee optimality under a thermal constraint since each core's power is only affected by its own speed. The temperature however, is affected by speeds of all cores. Moreover, even when thermal constraints were considered [5]–[7], the speed of each core was determined by an independent PID controller. This is, in general, sub-optimal as such a controller can only locally track a reference temperature threshold. It cannot perform a global optimization, as this requires predicting the temperature response of each core into the future. Further, such controllers cannot easily adapt when the load on the processor changes suddenly, as it requires changing their plant models.PID controllers for maximizing processor throughput under DTM were proposed in [8] for a single core processor. It can be shown that for a single speed control, minimizing the temperature error $T - T_{\max}$ also maximizes throughput.

For a multi-core processor with identical cores and threads, [9] assumed a single global speed control, and solved for the optimal speed s^* by setting $T(s^*) = T_{\max}$. However, the threads running on different cores usually have different power characteristics, and the cores themselves may dissipate different amounts of power due to process variations [10] and leakage dependence on temperature (LDT). Further, the threads can have different instructions per clock (IPC). The instruction throughput is then a weighted sum of the clock speeds. Additionally, one could assign weights to threads based on priority, or any other performance criterion. Hence, the optimal core speeds are, in general, not identical, and can only be computed by modeling the effect of each of them on the temperature of all other cores. In general, this requires robust optimal multi-input multi-output (MIMO) control techniques like model predictive control, which

978-1-4244-2820-5/08/$25.00 ©2008 IEEE

combines the optimizing abilities of an open-loop control scheme with the robustness that comes from feedback. In this work, we focus on developing efficient open-loop techniques to solve the multi-core speed optimization problem (let's call it problem P_1). In Section V-D, we discuss how such a scheme could be combined with feedback.

In [11], problem P_1 was solved using convex optimization techniques, but takes a few hours to compute [11], and is hence computed offline for a given MPSoC workload. A heuristic for P_1 was proposed in [12], which assumes that only a single core is at the maximum temperature, and then attempts to calculate the amount by which to reduce its speed or that of its neighbors to meet thermal constraints. But, with each core trying to achieve maximum performance, it is very likely that more than one core is at the maximum temperature. Further, both of these works did not model the non-linear dependence of leakage on temperature. Assuming leakage values at maximum temperature significantly underpredicts the optimal throughput. Analytical solutions for optimal speed control of a single-core processor, subject to thermal constraints, have been proposed in [13]–[15]. Our work addresses the problem for multi-core processors and numerically computes the speed curves of each core to maximize a weighted sum of the integrals of the speed curves over a given time window. Also, [13], [14] used a very simple lumped RC thermal model, and [15] used a model that ignored lateral thermal resistances and lumped the package. Our work uses the Hotspot thermal model, and yet, in some cases, provides analytical solutions.

B. Main contributions

- We formulate the steady-state multi-core performance optimization problem under thermal constraints as a linear program. Our thermal model is the Hotspot equivalent circuit model, and our power model uses a linear approximation for LDT. We show that the solution is such that the thermal budget is fully utilized among a subset of threads, while the others are assigned zero speeds. Hence, we impose a "fairness" constraint that ensures that each active core has a minimum speed equal to a fraction of the mean of all other active cores.
- We solve the local transient version of the above problem as a linear program. Here, the objective is to maximize the performance subject to thermal constraints over a small time interval (of the order of the die thermal time constant). We show that the solution to the above problem tends to be determined purely by the thermal constraint, and not the weights in the objective function and explain why. We show how the thermal constraint equations can be solved analytically, and this provides an efficient and accurate online solution technique.
- To address the limitation of the local solution of being insensitive to weights, we propose an exponentially decreasing speed curve that goes from full speed towards the steady-state solution determined before. We solve a non-linear program (NLP) to determine the rates of decrease of the exponential curve for each core. The NLP achieves maximum throughput over a given time window (which is of the order of the package time constant) subject to thermal constraints. To solve the NLP online, we propose an approximate version of the Hotspot model, which makes it possible to express the die temperatures as an analytical function of the time-varying speed curves.
- We propose a simple feedback controller to demonstrate how the proposed open-loop solutions can be used together with feedback to provide a solution that is robust to model error.
- We implement the proposed speed control policies in Matlab using data from Hotspot and PTScalar for our power and thermal models. We show that the local solution can be computed within 1 ms, and the global solution, within 1 s, for a processor with nine cores. We found that for cases with large variance in the objective function weights, the global solution achieves up to 4% larger throughput than the local solution. The open-loop solutions required less than 10% correction from the feedback controller due to model approximations.

II. Models, assumptions, and notation

A. Performance model

Consider a multi-core processor with n cores. We assume each core i can independently control its speed s_i. Here, speed refers to the DTM control mechanism like the clock gating duty cycle, fetch throttling duty cycle, clock frequency, etc. For these mechanisms, the dynamic power is a linear function of the speed. We do not model dynamic voltage scaling in this work for the following reasons: (i) the majority of multi-core processors today use fetch throttling and clock gating for DTM as they can be activated with smaller overhead, (ii) the margin for voltage scaling is already small and supply voltages are not expected to scale much in future technology generations [16]. We assume the speed can be varied continuously over $[0,1]$.

For simplicity, we assume each core can only run a single thread at a time (i.e. no SMT). For processors with many cores, having simple single-context cores is expected to be more power efficient [1]. We also assume that a thread runs on the same core it was initially assigned to until it completes. A core is said to be active if it is executing a thread and inactive otherwise. Inactive cores are assumed to be placed in a low power mode rather than simply idling. Given a set of n_t threads, the length n core activity vector $\mathbf{a}$ defines the mapping of threads to cores. If $a_i > 0$, the thread whose index is a_i is mapped to core i. If $a_i = 0$, core i is inactive. We define the throughput of a multi-core processor as a weighted sum of the speeds of each core $S = \sum_{i=1}^{n} w_i s_i$. For an active core i, the weight is a positive value that can indicate the average instructions per clock (IPC), priority, or any other performance criterion that is different for different threads. For an inactive core i, $w_i = 0$. For IPC weights, the above weighted sum is the number of instructions per second.

B. Power and thermal models

A given combination of core speeds (denoted by a vector $\mathbf{s}$) will result in a corresponding spatial and temporal distribution of power consumption. The spatial variations at a given time instant are due to differences in circuit type, size, activity, and even temperature among functional units and cores. The temporal variations on a given core are due to the nature of the code being executed (CPU-bound vs memory-bound), or a context switch. Denoting the power and temperature of all functional units on the chip as $\mathbf{P}_c$ and $\mathbf{T}_c$, we note that $\mathbf{P}_c = f(\mathbf{s}, \mathbf{T}_c, t)$, where the dynamic power depends on the speed and time, and the static power only on the temperature. We use the non-linear temperature-dependent leakage models from [17] to model $\mathbf{P}_c(\mathbf{T}_c)$. We compute a linear upper bound and use these models to find the open-loop speed control vectors. The feedback controller however, uses a plant model with non-linear leakage-temperature models, and can correct errors resulting from the linear approximation.

The temperature vector is related to the power vector through a dynamic relationship that requires knowledge of the thermal interface material, package, and convection vectors (collectively called $\mathbf{T}_p$) in addition to $\mathbf{T}_c$. We note that these non-die blocks dissipate zero power. Now, representing the combined chip-package power and temperature vectors as $\mathbf{P}$ and $\mathbf{T}$, respectively, we can describe their relation as a state-space differential equation [18], [19] $d\mathbf{T}/dt =$

$\mathbf{A}\ \mathbf{T}+\mathbf{B}\ \mathbf{P}(\mathbf{s},\mathbf{T},t)$. We use the Hotspot thermal equivalent circuit model [18] to obtain the state-space matrices $\mathbf{A}$ and $\mathbf{B}$ for a given multi-core floorplan. For a floorplan with m functional units per core, the Hotspot circuit has nm nodes/blocks each, in the die and thermal interface material layer (TIM), and 14 blocks in the package, for a total of $N = 2nm+14$ blocks. Note that our thermal system has N temperature states, and N power inputs (of which only nm are non-zero). Hence, both $\mathbf{A}$ and $\mathbf{B}$ are $N \times N$.

C. The system power vector

Assuming a linear approximation for LDT, we now express the system power vector $\mathbf{P}$ as a linear function of the speed vector $\mathbf{s}$ and the temperature vector $\mathbf{T}$. The dynamic power of each block when its corresponding core is running at full speed is defined as $\mathbf{P}_d^{(\max)}$. This vector is of size N, and consists of zeros for the $nm+14$ package and TIM blocks. It is also zero for any chip blocks in inactive cores. The actual dynamic power for chip block i is then given by $P_{d,i} = P_{d,i}^{(\max)} \sum_{i=1}^{n} x_{ij} s_j$, for all $i = 1,\ldots,nm$, and $j = 1,\ldots,n$, where $x_{ij} = 1$ if block i belongs to core j, and 0 otherwise. For $i > nm$, $x_{ij} = 0$ for all j. This can be expressed in matrix notation as $\mathbf{P}_d(\mathbf{s}) = \text{diag}\left(\mathbf{P}_d^{(\max)}\right)\ \mathbf{X}\ \mathbf{s}$, where $\mathbf{X}$ is the matrix of x_{ij} values and is of size $N \times n$, and the diag() of a vector $\mathbf{v}$ is diagonal matrix whose diagonal elements are equal to $\mathbf{v}$.

The leakage power vector for chip units can be expressed as $\mathbf{P}_{c,s} = \mathbf{P}_{c,s,0} + \mathbf{G}_{c,s}\mathbf{T}_c$. Here, $\mathbf{G}_{c,s}$ is a diagonal matrix with the non-zero elements representing the slopes of the power-temperature curve, and having units of thermal conductance. The leakage power and slope values correspond to active leakage for blocks in active cores and standby leakage otherwise. Padding the length nm power vectors and the slope matrix matrix with zeros (since non-die blocks dissipate no power), we get the length N system leakage power vector $\mathbf{P}_s = \mathbf{P}_{s0} + \mathbf{G}_s\mathbf{T}$. We then have the new system thermal equation $d\mathbf{T}/dt = \hat{\mathbf{A}}\ \mathbf{T} + \mathbf{B}\ (\mathbf{P}_d(\mathbf{s}) + \mathbf{P}_{s0})$, where $\hat{\mathbf{A}} \triangleq \mathbf{A} + \mathbf{B}\ \mathbf{G}_s$. This presents the power-thermal relationship once again in the standard state space form.

III. Problem formulation

A. The steady-state speed control problem

Given a multi-core processor with n_t threads, a weight vector $\mathbf{w}$, power coefficient vectors $\mathbf{P}_d^{(\max)}, \mathbf{P}_{s0}, \mathbf{G}_s$, find the speed vector $\mathbf{s}$ that maximizes the steady-state throughput $\mathbf{w}'\ \mathbf{s}$ (where $\mathbf{w}'$ denotes the transpose of $\mathbf{w}$) subject to the constraint that the steady-state temperature of each die block is no greater the threshold temperature $T_{\max}$, and that the core speeds are all in $[0,1]$.

$$\max_{\mathbf{s}_{ss}} \quad \mathbf{w}'\ \mathbf{s}_{ss}, \tag{1}$$

$$\text{s.t.} \quad d\mathbf{T}_{ss}/dt = 0 = \hat{\mathbf{A}}\ \mathbf{T}_{ss} + \mathbf{B}\left(\mathbf{P}_d(\mathbf{s}_{ss}) + \mathbf{P}_{s0}\right), \tag{2}$$

$$T_{i,ss} \le T_{\max}\ \forall\ i \in \{1,\ldots,nm\}, \tag{3}$$

$$0 \le \mathbf{s}_{ss} \le 1. \tag{4}$$

It can be shown that the state matrix $\hat{\mathbf{A}}$ is always invertible, so that we can write $\mathbf{T}_{ss} = -\hat{\mathbf{A}}^{-1}\mathbf{B}\left(\mathbf{P}_d(\mathbf{s}_{ss}) + \mathbf{P}_{s0}\right)$. Now, $\mathbf{P}_d(\mathbf{s})$ is a linear function. Selecting the first nm rows from the above equation, we can express (2) and (3) together as a linear constraint of the form $\mathbf{C}\mathbf{s}_{ss} \le \mathbf{d}$. We then obtain a linear program with n decision variables, nm linear constraints, and $2n$ simple bounds on the decision variables.

B. The transient speed-control problem

The transient speed control problem involves finding the time-varying speed curves of each processor to maximize the average throughput over a finite time window subject. It is subject to constraints on the die temperature at speed at every time instant.

$$\max_{\mathbf{s}(t)} \quad \frac{1}{t_f}\int_0^{t_f} \mathbf{w}'\ \mathbf{s}(t)\ dt, \tag{5}$$

$$\text{s.t.} \quad d\mathbf{T}(t)/dt = \hat{\mathbf{A}}\ \mathbf{T}(t) + \mathbf{B}\left(\mathbf{P}_d(\mathbf{s}(t)) + \mathbf{P}_{s0}\right), \mathbf{T}(0) = \mathbf{T}_0 \tag{6}$$

$$T_i(t) \le T_{\max}\ \forall\ i \in \{1,\ldots,nm\}, 0 \le t \le t_f, \tag{7}$$

$$0 \le \mathbf{s}(t) \le 1\ \forall\ 0 \le t \le t_f. \tag{8}$$

This is a problem in optimal control [20], and can only be solved numerically using techniques like dynamic programming. Due to the large control space (n speed variables) and state space (N temperature variables), such an approach would quickly explode, and would be unsuitable even for an offline computation. We present two strategies that are more amenable for online computation.

1) Local solution: Here, the time window $[0,t_f]$ is divided into time steps whose length t_s is of the order of the die thermal time constant. The speed curve for each core is assumed to be constant over this time period. Further, it is assumed that the temperature response to this step speed vector is monotonic over each time step. Since the initial temperature vector is guaranteed to be thermally feasible, we then only need to ensure that the temperature at the end of the time step meets the thermal constraint. We can formally state the local transient problem over the time duration $((k-1)t_s, kt_s]$ as

$$\max_{\mathbf{s}} \quad \mathbf{w}'\ \mathbf{s}, \tag{9}$$

$$\text{s.t.} \quad \mathbf{T}(kt_s) = e^{\hat{\mathbf{A}}t_s}\mathbf{T}((k-1)t_s) + \hat{\mathbf{A}}^{-1}\left(e^{\hat{\mathbf{A}}t_s} - \mathbf{I}_{N\times N}\right)\mathbf{B}\left(\mathbf{P}_d(\mathbf{s}) + \mathbf{P}_{s0}\right), \tag{10}$$

$$T_i(kt_s) \le T_{\max}\ \forall\ i \in \{1,\ldots,nm\} \tag{11}$$

$$0 \le \mathbf{s} \le 1, \tag{12}$$

where $\mathbf{I}_{N\times N}$ is the identity matrix. In (10), we have used the matrix exponential $e^{\hat{\mathbf{A}}t_s}$ to analytically solve the differential equation system (6) over the k^{th} time step. The matrix exponential is fixed for a given set of threads allocated to a given set of cores, and hence does not need to be frequently recomputed. As with the steady-state solution, we can show that (10) and (11) constitute a set of nm linear constraints on the speed vector $\mathbf{s}$, so that the above problem reduces to a linear program of the same size as that of the steady-state problem.

2) Global solution: The local solution operates over short time durations to ensure the monotonicity of the temperature response. But this is sub-optimal solution over the long-term. The global solution computes in advance the speed curve over a large time window. Although this could be solved as a large linear program, it would have $n_t n_s$ unknowns, where n_s is the number of time steps (which could be as large as 10,000). A recent work on single-core speed control on thermal constraints [15] showed that the optimal speed curve has the following property. It operates at the maximum speed until the temperature reaches the threshold. Then, it uses an exponential curve to asymptotically approach the steady-state thermally feasible speed.

Based on the above insight, we assume that the optimal speed curve for multi-core processors operates all cores at maximum speed until one of them reaches the temperature threshold. Once this happens, the speeds of all cores are exponentially decreased towards the steady solution $\mathbf{s}_{ss}^*$. Supposing one of the cores reaches thermal emergency at time 0, we then have for an active core i, $s_i(t) = s_{i0}e^{-t/\tau_i} + s_{i,ss}^*[1 - e^{-t/\tau_i}]$, where the τ_i's are unknown parameters. We denote this dependence on t and τ as $\mathbf{s}(\tau,t)$. We then solve the following NLP to determine these parameters, and impose the thermal

constraints at several time instants $t_1, t_2, \ldots, t_p$.

$$\max_{\tau} \quad f(\tau) = \frac{1}{t_f}\int_0^{t_f} \mathbf{w}'\ \mathbf{s}(\tau,t)\ dt, \tag{13}$$

$$\text{s.t.} \quad \mathbf{T}(t_k) = \mathbf{g}(\mathbf{T}_0, \tau, t_k) \tag{14}$$

$$T_i(t_k) \le T_{\max}\ \forall\ i \in \{1,\ldots,nm\}, k \in \{1,\ldots,p\}. \tag{15}$$

The objective function can be expressed in closed-form as $f(\tau) = \sum_{i=1}^{n}\left[w_i\left(s^*_{i,ss}t_f + \left(s_{i,0} - s^*_{i,ss}\right)\left(1-e^{-t_f/\tau_i}\right)\right)\right]$. In the extended version of this paper, we present an approximate thermal model where the temperature vector can be expressed analytically as a function of the unknown parameters τ. This allows us to solve the above NLP efficiently (in about one second).

IV. Properties of the thermal constraint matrix

A. Constrain only the hottest block in each core

Since the thermal upper bound is the same for all die blocks, we are effectively constraining only the hottest die block(s). If we could identify these blocks beforehand, we can retain only the corresponding rows in the constraint matrix. This can be done by profiling workloads and observing the set of (say) k hottest blocks. In practice, this set is small. For example, [8] found that the integer register was the hottest unit for a number of SPECInt benchmarks. This way, we can reduce the number of thermal constraint equations to as low as n_t (from nm). These can be compactly written as $\mathbf{us} + \mathbf{v} \le T_{\max}$, where $\mathbf{u}$ and $\mathbf{v}$ are of size $n_t \times n$ and $n_t \times 1$.

B. Global and local components of the constraint equations

We now use an approximate power and thermal model to show that each row of the matrix $\mathbf{u}$ can be divided into a global component (common to all rows) and a local component (specific to each row). Figure 1 shows a simple thermal equivalent circuit. Each block i on the die is modeled as a current source p_i (that represents its power consumption) and a thermal resistance $R_{\text{die,i}}$. Let the block power be expressed as the sum of dynamic and static components $p_i = p^{(\max)}_{d,i} s_{c_i} + p_{s,i}$, where c_i is the index of the core to which block i belongs. Each die block has a corresponding thermal interface material (TIM) block, which is modeled as another thermal resistance $R_{\text{int,i}}$. The package and cooling system blocks are modeled as a single lumped thermal resistance R_p. Let us define $R_i \triangleq R_{\text{die,i}} + R_{\text{int,i}}$, and let C_k be the set of blocks that belong to core k. The total dynamic power of core k is given by $P_{d,k} = s_k P^{(\max)}_{d,k}$, where $P^{(\max)}_{d,k} = \sum_{i\in C_k} p^{(\max)}_{d,i}$. Similarly, the total static power of core k is given by $P_{s,k} = \sum_{i\in C_k} p_{s,i}$.

$$T_i = \left(p^{(\max)}_{d,i} s_{c_i} + p_{s,i}\right) R_i + R_p \sum_{k=1}^{n}\left(P^{(\max)}_{d,k} s_k + P_{s,k}\right) \tag{16}$$

$$= u^{(\text{local})}_i s_{c,i} + \sum_{k=1}^{n} u^{(\text{global})}_k s_k + v_i, \tag{17}$$

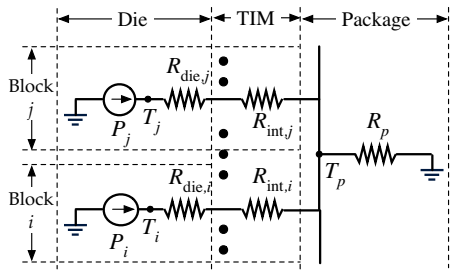

Fig. 1. Thermal model that shows global/local temperature components.

where $u^{(\text{local})}_i = p^{(\max)}_{d,i} R_i$, $u^{(\text{global})}_k = P^{(\max)}_{d,k} R_p$ and $v_i = p_{s,i} R_i + R_p \sum_{k=1}^{n} P_{s,k}$. The local component $u^{(\text{local})}_i s_{c,i}$ depends only on the speed of the local core s_{c_i}, while the global component $\sum_{k=1}^{n} u^{(\text{global})}_k s_k$ is independent of i, and is hence the same for all cores.

We numerically verified that for a quad-core processor thermal model, the $\mathbf{u}$ vector can be approximated as the sum of a diagonal matrix and a matrix with identical rows as suggested above. Such an approximation resulted in a worst-case error of at less than 1.6% and 1.9% respectively, for the steady-state and transient problems. Further, we observed that the for the transient problem, the non-diagonal elements of $\mathbf{u}$ (and hence the elements of $\mathbf{u}^{(\text{global})}$ are much smaller than for the steady-state case. This is because the package temperature does not vary significantly over the local solution's time step. So, the global component is not strongly dependent on the speed.

V. Efficient solution techniques

A. The steady-state solution and fairness

By constraining only the hottest units, the size of the linear program presented in Section III-A reduces considerably. Further, as the steady-state solution is computed only about once every 100 s as part of the global solution, the computation time is not as critical as it is for the local solution. However, we now show that the optimal solution to the steady-state problem has an undesirable property.

Lemma 1: Let $T_{i,ss}$ be the hottest die temperature of core i for the optimal steady-state solution. If $T_{i,ss} < T_{\max}$, $s^*_{i,ss}$ is either 0 or 1.
The proof is presented in the extended version of the paper. This result implies that the optimum steady-state solution selects a set of cores to operate at either full speed or at a speed that achieves the maximum temperature. It is possible that some of the active cores (with $T_{i,ss} < T_{\max}$) could be completely turned off, since no cores can exist with both a thermal slack and a speed slack. A thread with low weight could receive a steady-state speed of zero, because it is better (in terms of throughput) to allocate the available thermal budget among higher weight threads. In fact, this could occur even when all cores have equal weight if the cores have sufficiently different power characteristics. Then, the solution tends to prefer lower power threads.

To avoid this, we propose the following "fairness" constraint: for each active core i, the speed at any instant $s_i(t)$ must be no less than a certain fraction f of the mean of the speeds of the other active cores. The addition of this constraint results in a solution with lower throughput. By adjusting the value of f over $[0,1)$, we can achieve a trade-off between fairness and throughput.

B. Analytical solution to the local transient problem

The problem formulation in Section III-B.1 requires solving a linear program with n_t decision variables, and about $3n_t$ constraints. This is still inconvenient for repeated online computation. We now show that the optimal solution to this problem is determined completely by the constraint matrix, so that it is sufficient to solve the corresponding set of linear equations to determine the optimal solution.

Lemma 2: Let $T_i(t_s)$ be the hottest die temperature of core i at the end of the time step t_s. For time steps of the order of the die thermal time constant or less, we have that if $T_i(t_s) < T_{\max}$, then $s^*_i = 1$.
The proof is presented in the extended version of the paper. As a consequence of this result, the optimum solution to the local transient problem selects a set of active cores to operate at maximum die temperature. Any other active core operates at full speed. This way, the only unknown speeds are of those cores that have binding thermal constraints. One can then compute the optimum speed vector by assuming all cores are at the maximum temperature, and if the resulting speed of any core exceeds 1, it is simply set to 1.

The thermal constraint equations can be written as $\mathbf{us} + \mathbf{v} \leq T_{\max}$. We can also simplify $\mathbf{u} \approx \mathbf{u}^{(\text{local})} + \mathbf{u}^{(\text{global})}$. Since, the local component is a diagonal matrix and the global one has identical rows, we can denote $u_i^{(\text{local})} \triangleq \mathbf{u}_{i,i}^{(\text{local})}$ and $u_i^{(\text{global})} \triangleq \mathbf{u}_{j,i}^{(\text{global})}$ $\forall\ j$. With some algebra, it can be shown that the above set of equations can be solved analytically to obtain the optimum speeds of each core as follows. First, we compute the speed of an arbitrary reference core r as

$$s_r = \frac{T_{\max} - v_r + \sum_{i=1}^{n} a_i (v_i - v_r) \left(u_i^{(\text{global})} / u_i^{(\text{local})} \right)}{u_r^{(\text{local})} \left[1 + \sum_{i=1}^{n} a_i \left(u_i^{(\text{global})} / u_i^{(\text{local})} \right) \right]}, \qquad (18)$$

where $a_i = 1$ if core i is active, and zero otherwise. Then, the speeds of the other active cores are computed in terms of s_r as

$$s_i = \left[u_r^{(\text{local})} s_r - (v_i - v_r) \right] / u_i^{(\text{local})},\ \forall\ i \in \{1, \ldots, n\}, \text{and } a_i = 1. \quad (19)$$

Finally, we do $s_i^* = \min(s_i, 1)$ for all active cores.

C. The global transient solution

We formulated the global transient speed control problem as an NLP in Section III-B.2. To solve it efficiently online, the thermal constraints must be expressed analytically in terms of the decision variables τ. This is not possible with the Hotspot thermal model. However, if we can reduce the number of state variables from N temperatures to a single temperature, we can indeed compute the response analytically. Now, the heatsink-convection equivalent capacitance $C_{\text{hs-conv}}$ is more than 10 times larger than those in the spreader, TIM, and die. So, we lump the resistances and capacitances in the heatsink-convection layer, and neglect all other capacitances.

Since the global solution only needs to constrain the temperature at times of the order of seconds, neglecting these faster dynamics introduces little error. The detailed derivation of the die temperature as a function of both, the time and the parameters of the speed curve τ, can be found in the extended version of the paper. We verified numerically that for a quad-core processor thermal model, the worst-case and mean transient prediction errors (of the approximate model w.r.t. the Hotspot model) are about 9.0°C and 2.7°C respectively, and the steady-state error is about 1°C. These errors are sufficiently small that they can be corrected using a feedback controller.

D. A simple feedback controller

The above open-loop solutions assume a linear approximation for LDT, a simplified thermal model, or both. Further, it is difficult to obtain power information at the functional unit level, and even detailed thermal models could be inaccurate. Hence, if the proposed open-loop solutions were directly implemented in a real multi-core system, we could have: (i) a temperature underprediction, which means there is thermal slack available for one or more cores to increase their speeds, or (ii) a temperature overprediction, which will activate the processor's fall-back throttling mechanism; this is usually a fast acting but less power-efficient technique like clock gating.

We propose a simple feedback mechanism shown in Figure 2 to illustrate how feedback can help avoid the effects of temperature misprediction, but retain the optimizing ability of the open-loop control. The controller scales the open-loop solution s_{OL}^* by a scalar value k to minimize the error between the maximum measured die temperature and the thermal threshold. This factor k is found by performing a binary search over the typical operating range of this parameter (around $[0.5, 1.5]$). In practice, more sophisticated controllers (ex. PID control, MIMO control, model predictive control) will be required and offer potentially higher throughputs, but they are beyond the scope of this paper. We use the control scheme proposed in Figure 2 to demonstrate that the amount of correction (quantified by k) required is less than $\pm 10\%$ for the proposed open-loop schemes.

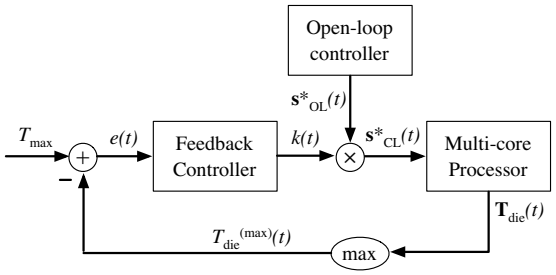

Fig. 2. Feedback controller used in combination with open-loop control.

Benchmark	bzip2	crafty	gcc	twolf
IPC	1.05	0.81	0.56	1.52
Total dynamic power (normalized)	1.00	0.98	1.00	0.86
Total static power (normalized)	0.88	0.88	1.00	1.00
Speed solution at end of time window using IPC weights				
s^*_{local}	0.73	0.60	0.59	0.69
s^*_{global}	0.95	0.22	0.22	0.95
Throughput relative to single core at full speed with IPC = 1				
Throughput (local)	2.72			
Throughput (global)	2.83 (4% improvement)			

TABLE I
BENCHMARK STATISTICS AND MULTI-CORE SPEED SOLUTION.

VI. Experimental results

A. Simulation setup

Our work focuses on the power/thermal behavior of multi-core processors. Simulating them at the cycle-accurate level to capture long-term thermal behaviors is very time-consuming (several hours or days). Hence, our approach is to perform simulation at a granularity of around 10 ms. This is of the order of the die time constant, and the operating system scheduler time slice, and is the least time step at which power and temperature change significantly. We use the block-based model in Hotspot 4 [18] as our thermal model with a multi-core version of the Alpha 21264 floorplan. For a given floorplan, Hotspot computes the state space matrices $\mathbf{A}$ and $\mathbf{B}$ described in Section II-B at the functional unit level. We choose the thermal threshold $T_{\max}$ as 110°C, and set the convection thermal resistance in Hotspot to 0.35 °C/W. We obtain power traces for different SPEC CPU2000 benchmarks using PTScalar [17], which is a cycle-accurate power simulator which outputs dynamic power and temperature-dependent leakage power values based on 65 nm technology. These power traces are then scaled down depending on the number of cores used in the simulation. We implement our control strategies in Matlab.

B. Comparison of (temporally) global and local control techniques

Previous studies like [5], [7] have established that having (spatially) local control over core speeds provides greater throughput than having a single global control. Here, we compare the benefit of using a long-term solution (which accounts for weights in the throughput function) versus a short-term solution (which only cares about thermal constraints). Table I shows a set of four SPEC benchmarks, and statistics related to their IPC's and their relative power consumption.

Each of these threads was mapped to a separate core of a quad-core processor. We then computed the local and global transient solutions using IPCs as weights for a time window of 300 s. The resulting speed curves are shown in Figures 3 and 4. Initially, all cores run at full

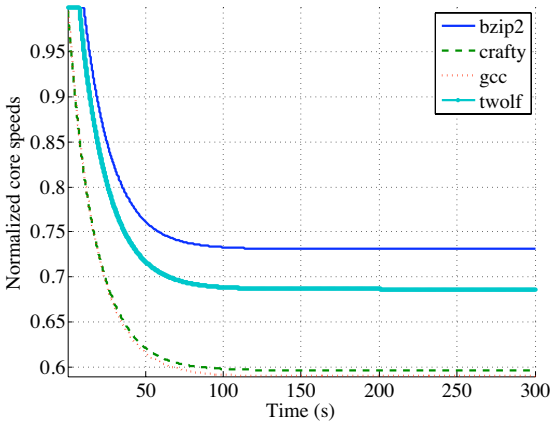

Fig. 3. Local transient speed solution with IPCs as weights.

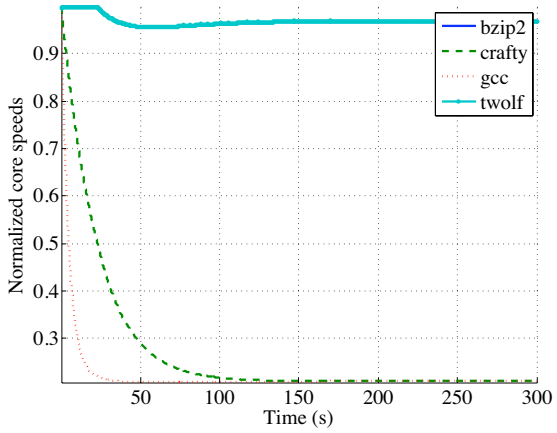

Fig. 4. Global transient speed solution with IPCs as weights.

speed. At time $t = 0$, one of the cores hits $T_{\max}$. The global solution achieved a 4% greater throughput (relative to a single core operating at full speed with an IPC of 1) than the local one. The local solution differentiates between cores based only on their powers, whereas the global solution also considers the IPC weights.

Note that the global speed curve is not monotonically decreasing as assumed. This is because of the action of the feedback controller, which corrects for the temperature misprediction due to the approximate thermal model. The resulting correction was less than 5%. Also, the global speed curve runs gcc and crafty at a low speed of around 0.22 in steady state. This is because we used a fairness constraint with $f = 0.3$, so these threads got at least 30% of the mean of all other threads. Without these constraints, their steady-state speeds would have been zero. The local solution is computed in 1 ms per time step. We believe an optimized implementation in C can be made to run within 0.1 ms. The global solution requires about 3 s to compute the global solution and solve the NLP. By restricting this computation to every 100 s, the overhead is about 3%, which can again be reduced with an optimized C implementation.

VII. Conclusion

Dynamic thermal management will play a dominant role in the operation of future many-core processors due to the larger operating range of die temperatures, and the high cost of worst-case thermal design. To maximize throughput under thermal constraints, multi-core's must use speed controllers that combine open-loop optimizing controllers with robust feedback controllers. This allows them to intelligently allocate the available thermal budget among non-identical cores and threads. We formulate linear and non-linear program formulations for this open-loop problem. We present theoretical results and practical observations that help reduce the complexity of these solutions. We demonstrate that our solutions can be computed in times suitable for online implementation, and with little corrections from the feedback controller. The global solution provides a small improvement in throughput over the local solution with little additional cost. The improvements increase with the variance in the performance weights among threads. The proposed techniques are the first to address this problem with online solutions while accounting for leakage dependence on temperature and thermal transients.

References

[1] S. Borkar, "Thousand core chips – A technology perspective," in *Proc. Design Automation Conf. (DAC)*, June 2007, pp. 746–749.
[2] S. Siddha, V. Pallipadi, and A. Mallick, "Process scheduling challenges in the era of multi-core processors," *Intel Technology Journal*, vol. 11, no. 4, pp. 361–369, November 2007.
[3] J. Li and J. F. Martínez, "Power-performance considerations of parallel computing on chip multiprocessors," *ACM Trans. Archit. Code Optim.*, vol. 2, no. 4, pp. 397–422, 2005.
[4] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, "An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget," in *Proc. Intl' Symp. Microarch. (MICRO)*, 2006, pp. 347–358.
[5] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," in *Proc. ISCA*, 2006.
[6] Y. Li, B. Lee, D. Brooks, Z. Hu, and K. Skadron, "CMP design space exploration subject to physical constraints," in *Proc. Intl' Symp. High Perf. Comp. Arch. (HPCA)*, 2006, pp. 15–26.
[7] P. Chaparro, J. González, G. Magklis, Q. Cai, and A. González, "Understanding the thermal implications of multicore architectures," *IEEE Trans. Parallel and Distributed Sys.*, vol. 18, no. 8, pp. 1055–1065, August 2007.
[8] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM TACO*, vol. 1, no. 1, pp. 94–125, 2004.
[9] R. Rao, S. Vrudhula, and C. Chakrabarti, "Throughput of multi-core processors under thermal constraints," in *Proc. Intl' Symp. Low Power Electronics and Design (ISLPED)*, August 2007, pp. 201–207.
[10] J. Donald and M. Martonosi, "Power efficiency for variation-tolerant multicore processors," in *Proc. Intl' Symp. Low Power Electronics and Design (ISLPED)*, 2006, pp. 304–309.
[11] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, and G. D. Micheli, "Temperature-aware processor frequency assignment for MP-SoCs using convex optimization," in *Proc. Intl' Conf. Hardware Software Codesign (CODES)*, 2007, pp. 111–116.
[12] R. Mukherjee and S. O. Memik, "Physical aware frequency selection for dynamic thermal management in multi-core systems," in *Proc. Intl' Conf. Computer Aided Design (ICCAD)*, 2006, pp. 547–552.
[13] A. Cohen, F. Finkelstein, A. Mendelson, R. Ronen, and D. Rudoy, "On estimating optimal performance of cpu dynamic thermal management," *IEEE Computer Architecture Letters*, vol. 2, no. 1, pp. 6–6, 2003.
[14] S. Zhang and K. Chatha, "Approximation algorithm for the temperature-aware scheduling problem," in *Proc. ICCAD*, November 2007.
[15] R. Rao and S. Vrudhula, "Peformance optimal processor throttling under thermal constraints," in *Proc. CASES*, October 2007, pp. 257–266.
[16] P. Clarke, "Physics is not the hurdle for scaling CMOS," *EE Times*, 2003. [Online]. Available: http://www.eetimes.com/story/OEG20030923S0037
[17] W. Liao, L. He, and K. M. Lepak, "Temperature and supply voltage aware performance and power modeling at microarchitecture level," *IEEE TCAD*, vol. 24, no. 7, pp. 1042–1053, July 2005.
[18] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, and S. Ghosh, "Hotspot: A compact thermal modeling method for CMOS VLSI systems," *IEEE Trans. VLSI Sys.*, vol. 14, no. 5, pp. 501–513, May 2006.
[19] Y. Han, I. Koren, and C. M. Krishna, "Temptor: A lightweight runtime temperature monitoring tool using performance counters," in *Workshop on Temp. Aware Comp. Sys. (TACS)*, 2006.
[20] D. S. Naidu, *Optimal Control Systems*. CRC Press, 2002.