Stochastic Modeling of a Thermally-Managed Multi-Core System

Hwisung Jung
Department of EE-Systems
University of Southern California
Los Angeles, CA
hwijung@usc.edu

Peng Rong
ASIC Development
Brocade Communications Systems
San Jose, CA
prong@brocade.com

Massoud Pedram
Department of EE-Systems
University of Southern California
Los Angeles, CA
pedram@ceng.usc.edu

ABSTRACT
Achieving high performance under a peak temperature limit is a first-order concern for VLSI designers. This paper presents a new abstract model of a thermally-managed system, where a stochastic process model is employed to capture the system performance and thermal behavior. We formulate the problem of dynamic thermal management (DTM) as the problem of minimizing the energy cost of the system for a given level of performance under a peak temperature constraint by using a controllable Markovian decision process (MDP) model. The key rationale for utilizing MDP for solving the DTM problem is to manage the stochastic behavior of the temperature states of the system under online re-configuration of its micro-architecture and/or dynamic voltage-frequency scaling. Experimental results demonstrate the effectiveness of the modeling framework and the proposed DTM technique.

Categories and Subject Descriptors
D.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aides

General Terms
Algorithms, Design, Performance

Keywords
Dynamic thermal management, stochastic processes, uncertainty

1. INTRODUCTION
Ongoing advances in CMOS process technologies and VLSI designs have resulted in the introduction of high-performance multi-core systems on a chip (SoC). Thermal control in such systems has become a first-order concern due to the increased power density and thermal vulnerability of the chip. Localized heating is a frequent occurrence in SoC designs. Power dissipation is spatially non-uniform across the chip, resulting in emergence of hot spots and spatial temperature gradients that can cause accelerated aging, timing errors (setup time violations), or even physical damage to the chip. To solve this, dynamic thermal management (DTM) techniques, which attempt to ensure thermal safety by employing runtime mechanisms to control power density and to prohibit excessive local heating, have been proposed as a class of micro-architectural solutions and control strategies, which seek to enable the highest SoC performance while meeting peak temperature constraints.

As reported in [1]-[5], the problem of thermal modeling and management has received a lot of attention. The work presented in [1] relies on a compact thermal model to achieve a temperature-aware design methodology. A thermal control mechanism used to cool the microprocessor's temperature has been derived in [2]. Predictive thermal management [3], which exploits certain properties of multimedia applications, is an example of online strategies for thermal management. In [4], design guidelines for power and thermal management for high-performance microprocessors are provided. A summary of research that combine interconnect thermal effects and reliability measures is given in [5].

Much of the past work has examined techniques for thermal modeling and management, but these techniques may be ineffective to reduce chip temperature of multi-core (MC) systems because the configurability of the micro-architecture depending on the target application and the uncertainty in temperature measurement (erroneous or noisy temperature reports) have often not been considered. Furthermore, thermal models, based on equivalent circuit models, cannot adequately model heat generation and diffusion in structures with complex shapes and boundary conditions. Indeed, it is extremely difficult to obtain the exact solution of the heat equations that arise from realistic die conditions [6]. These difficulties render the problem of identifying hot spots stochastic.

In this paper, we present a stochastic model of a thermally-managed MC system (which we shall call **TMS**, for short) using a Markov decision process (MDP) model. Recall that MDP, which provides a robust theoretical framework for resource management problems, is a theory of modeling the sequential decision making process [7]. The key rationale for utilizing MDP for solving the DTM problem in MC systems is to manage the stochastic behavior of the temperature states of the system under dynamic re-configuration of its micro-architecture (which may take place in response to application program characteristics), while maximizing the system performance subject to the constraint that a critical temperature threshold is not exceeded locally or globally.

The remainder of this paper is organized as follows. Section 2 provides some preliminaries of the paper, while section 3 describes the details of the proposed models for a TMS. Section 4

This research is supported in part by the National Science Foundation under grant no. 0509564.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DAC 2008, June 8–13, 2008, Anaheim, California, USA
Copyright 2008 ACM 978-1-60558-115-6/08/0006…5.00

presents a DTM problem formulation. Experimental results and conclusions are given in section 5 and section 6.

2. PRELIMINARIES

A modern computing system, which typically utilizes multi-cores to achieve high performance, exhibits different thermal profiles under different application programs due to its re-configurable micro-architecture. For example, its cache size varies drastically, depending on the characteristics of the running threads (i.e., application programs), where these adaptive caches adjust from small sizes with fast access time to higher capacity but slower and more power hungry configurations. As expected, larger cache configurations, which are more prominent for dual-thread workloads, provide higher power dissipation than smaller size cache. This in turn dynamically changes the temperature profile of the SoC during program execution. Details of the functionality of the MC systems and algorithms for changing the micro-architectural configuration on-the-fly fall outside the scope of the present paper. Interested readers may refer to [8].

Application programs tend to exhibit different characteristics as a function of the program phase they are in [9]. This is in turn affects the computational workload of the processor, causes a new micro-architectural configuration to be employed, which in turn results in a different thermal profile on the chip. Figure 1 shows the obtained IPC (Instruction per Cycle) by running various application programs (e.g., SPEC CPU2000 [10]) on the Intel Core Duo processor with a typical architectural specification (cf. [11]). In this figure, IPCs for applications are compared to L2 cache miss rate, where average IPC for CPU2000 benchmarks is measured as 0.85. It is clearly seen that higher L2 cache miss rate accounts for its low IPC.

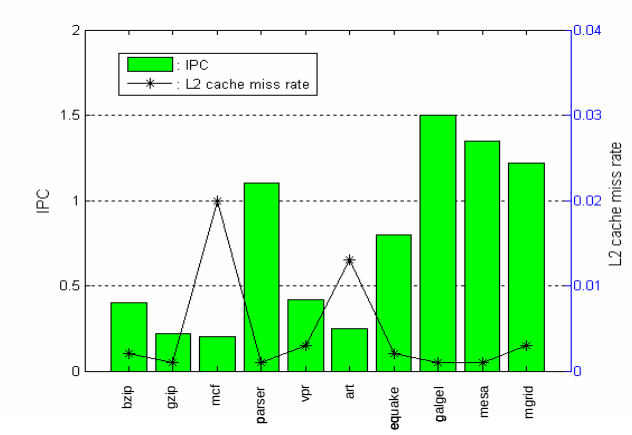


Figure 1. IPC vs. L2 cache miss rate on Intel Core Duo processor.

An integrated circuit (device) is typically allowed to operate when the ambient air temperature, T_A, surrounding the device package, is within the range of 0°C to 70°C [12]. It is expedient to define the *critical temperature threshold*, T_{crit}, as the temperature above which a chip is in thermal violation resulting in timing errors or accelerated device/interconnect aging, and a *trigger temperature threshold*, T_{trig}, as the temperature above which DTM techniques are employed. A thermal manager employs temperature reduction mechanisms when the system temperature exceeds a pre-defined temperature threshold (i.e., the trigger temperature).

Temperature reading can be performed by either external or internal thermal sensors. External thermal sensors, e.g., thermocouples, incur a rather large time delay in reading the temperature and tend to produce less accurate temperature measurements. Internal thermal sensors, e.g., solid state sensors, which may be deployed in larger numbers across a chip, have been developed in pursuit of higher accuracy. However, there still remain inaccuracies associated with the solid state sensors. For example, current biased temperature sensors are in general sensitive to noise on power and ground lines, and thus the sensor output for low temperature reading is affected by process variations, etc.

3. SYSTEM MODELING

We present a stochastic modeling technique to construct a TMS by utilizing a continuous-time Markov decision process (CTMDP).

3.1 Background

A CTMDP is a controllable continuous-time Markov process, which satisfies Markovian property [7] and takes a set of state $s \in S$, where state transition rates are controlled by actions $a \in A$. We consider a cost function which assigns a value to each state and action pair by adopting a conventional approach, i.e., when the system makes a transition from state s to another state s', it receives a cost.

Given a CTMDP with n states, its *generator matrix* **G** is defined as an $n \times n$ matrix, where an entry $\sigma_{s,s'}$ in **G** is called the *transition rate* from state s to another state s', which can be calculated as

$$\sigma_{s,s'}(a) = \delta(s', a) \cdot (1/\tau(s, s')), \quad s \neq s' \tag{1}$$

where $\tau(s, s')$ is a transition time from s to s', and $\delta(s', a)$ is 1 if s' is the destination state of action a or 0 otherwise. We can calculate the limiting distribution (steady) state probabilities of the CTMDP from its generator matrix. If state transition rates are controlled by actions chosen from a finite set of action A, a policy is defined as a set of state-action pairs for all the states of the CTMDP. The details of the CTMDP are omitted here. Interested readers may refer to [7].

The exponential distribution for state transition times, a prominent property of CTMDP model, is sometimes insufficient to model practical cases, especially when we model the first request arrival in the idle state period [13], where the inter-arrival times of service requests are in this case generally distributed. However, it will not hurt the quality of the present paper if we assume that the task inter-arrival times are exponentially distributed during the active state period since thermal management is only in effect during the program execution. Furthermore, the burst of program execution on a processor follows exponential distribution [14].

3.2 Component Models

We present a CTMDP-based model of a TMS to optimally solve the DTM problem. Figure 2 shows an abstract model of a TMS, which comprises of three components: processor, application program, and thermal sensor. In this paper, for simplicity we assume that each application is executed by one processor and that individual thermal sensors measure temperatures of each and every processor in the MC system. A new application may cause micro-architectural re-configuration of the corresponding processor in order to improve the overall performance. A thermal

manager (TM) receives state (phase) of the application, reads temperature data from the thermal sensor, and issues commands to the processor under its control to manage the temperature rise above T_{trig}. There is one TM assigned to each processor. Notice that R_i, S_j, and H_k represent the state sets of the application program (i = 1, 2, …, l), the processor (j = 1, 2, …, m), and the temperature (k = 1, 2, …, n), respectively, where l, m, and n are the number of applications, processors, and thermal sensors available within a MC system. Next, we construct the CTMDP model of a single processor system for simplicity. The CTMDP model of a MC system can be constructed in the same manner.

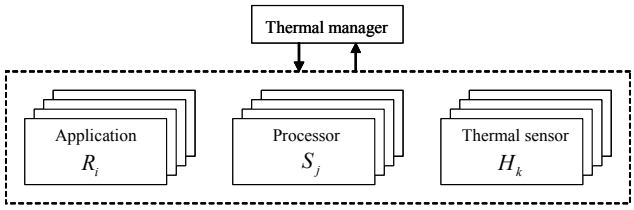


Figure 2. Abstract model of a thermal-managed MC system.

3.2.1 Modeling the Processor State

The CTMDP model of the processor is constructed as follows. Assume that each state $s \in S$ represents a combination of a micro-architectural configuration $c \in C$ (e.g., register file sizing, cache sizing, or float-point-unit disabling) and an action $a \in A$ (e.g., operating voltage-frequency (VF) setting), where there are micro-architectural configuration set $C = \{c_1, c_2, \ldots, c_u\}$ and action set $A = \{a_1, a_2, \ldots, a_v\}$ available to the processor. Thus, the CTMDP model of the processor includes a state set $S = \{s_1, s_2, \ldots, s_w\}$ and a parameterized generator matrix $\mathbf{G}_{\text{proc}}$, where w is the numbers of states of the processor, i.e., $w = u \cdot v$. A state transition out of some state s is controlled by either an action $a \in A$ or a configuration change $c \in C$. Any state transition takes a certain amount of time to complete, where this latency overhead ranges from several clock cycles to hundreds of milli-seconds. A typical micro-architecture re-configuration latency, the duration between the time a decision is made to change the micro-architectural configuration and the time of actual configuration, takes up to tens of clock cycles [16]. Thus, a state transition time in the CTMDP model of the processor takes $\tau(s, s')$ time (= max (τ_{DVFS}, τ_{ARCH})), where τ_{DVFS} is the transition time of dynamic voltage and frequency scaling (DVFS), and τ_{ARCH} is the micro-architecture transition period, when system transits from state s to s'.

An example of how to construct the CTMDP model of the processor is given next. For simplicity, we suppose that the processor has three micro-architectural configurations (e.g., cache resizing) denoted by c_1, c_2, and c_3, and a voltage frequency (VF) setting chosen from a finite set of actions $A = \{a_1, a_2, a_3\}$ is applied to the processor, where $a_1 < a_2 < a_3$ in terms of the VF values. Then, the abstract CTMDP model of the processor can be illustrated as shown in Figure 3 (a), where a node represents a processor state and a directed arc represents a transition between two states with the parameterized generator $\mathbf{G}_{\text{proc}}$. In Figure 3 (b), $\sigma_{s,s'} = \infty$ means the processor switches from state s to s' immediately (i.e., $s = s'$), and $\sigma_{s,s'} = 0$ means the processor can never switch from state s to s'. Note that upsizing cache configuration may require different time compared to downsizing the cache configuration. In addition, cache downsizing occurs at the beginning of the DVFS change because of the required cache partitioning time, whereas cache upsizing happens at the end of the DVFS change [15][16].

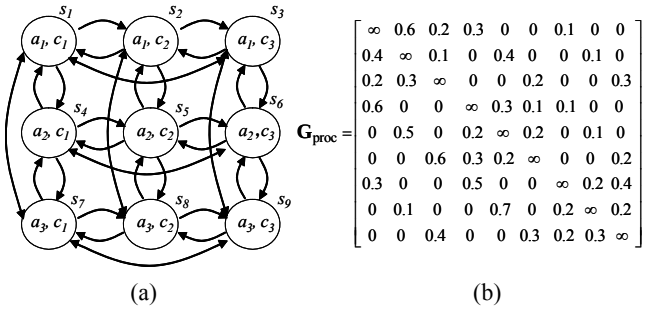


Figure 3. An example of CTMDP model of a processor (a) and its generator matrix (b).

3.2.2 Modeling the Application State

Application programs can be characterized by using their architecture-dependent characteristics (such as the IPC and cache-miss rate), architecture-independent characteristics (such as data and instruction temporal localities), or a combination of these two [17]. In this paper, we focus on the micro-architecture re-configurations that affect the IPC and data cache-miss rate characteristics of application programs, which subsequently result in temperature change on the processor die. Measuring the architecture-independent characteristics may be achieved by exploiting the notion of data similarity (e.g., instruction level parallelism, data locality). However, it is not straightforward to estimate the performance of a particular architecture-independent enhancement; therefore, we do not consider them here.

Inspired by these observations, we construct a CTMDP model of an application program. The CTMDP model consists of a state set $R = \{r_1, r_2, \ldots, r_p\}$ and a generator matrix $\mathbf{G}_{\text{app}}$, where p is the number of states that are present in the application. In our problem setup, application state r is differentiated based on values of IPC and the cache-miss rate. A state transition between different application states takes place autonomously, and may initiate a change in the state of the processor.

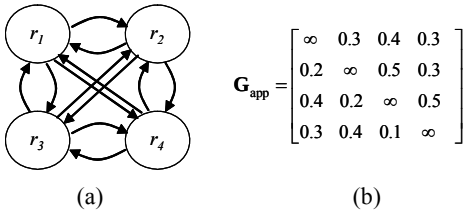


Figure 4. An example of CTMDP model of applications (a) and its generator matrix (b).

An example of a four-state CTMDP model of an application, considering workload characteristics, is depicted in Figure 4. Here r_1, r_2, r_3, and r_4 represent combinations of IPC and cache-miss rate (η) ranges of the application, e.g., r_1 = [IPC ≤ 0.85, $\eta \leq$ 0.01], r_2 = [IPC ≤ 0.85, $\eta >$ 0.01], r_3 = [IPC > 0.85, $\eta \leq$ 0.01], and r_4 = [IPC > 0.85, $\eta >$ 0.01]. Note that the threshold values for the IPC and η are set by the application developers. A state transition between different application states occurs within a processor. If we consider multiple processors with multiple applications, individual state transitions for each processor must

be considered. The transition rate $\sigma_{r,r'}$ in $\mathbf{G}_{app}$ includes the context switch time, not assuming a round-robin context switching architecture, controlled by the operating system. For example, if we make a context switch when the deadline for completing an application program is missed, then a state transition will occur with a specific transition rate.

3.2.3 Modeling the Temperature State

Temperature readings from thermal sensors are important to DTM technique, since by knowing the temperature profile of a chip, the TMS may be triggered to respond to chip temperature changes so as to avoid thermal failure/damage of the chip or to maximize performance of interest under temperature constraints.

Conventionally, the junction temperature T_J of the IC can be estimated with

$$T_J = T_A + P \cdot \theta_{JA} \tag{2}$$

where T_A is the ambient temperature (°C), P is the device power dissipation (W), and θ_{JA} is the thermal resistance from device junction to ambient (°C/W). In general, thermal failure is avoided by maintaining the device θ_{JA} value small enough so that the junction temperature T_J does not exceed a maximum value during operation. It is worthwhile to note that θ_{JA} cannot be modeled directly due to the complexity of thermal models for the package, cooling system, and board stack-up [6]. In addition, θ_{JA} is assumed to be a single parameter under the assumption that device power dissipation, P, is distributed uniformly across the die, which is not realistic assumption (i.e., uncertain behavior). To overcome this difficulty, we use an observation (i.e., temperature reading T_T of the package top obtained by a thermal sensor) as

$$T_J = T_T + P \cdot \psi_{JT} \tag{3}$$

where ψ_{JT} is the junction-to-top of package thermal characterization parameter used as a measure of the temperature difference between junction and package top surface, and is estimated from JEDEC thermal tests [12]. The device power P, a major source of heat generation, is varied based on micro-architectural configurations, which are also application dependent.

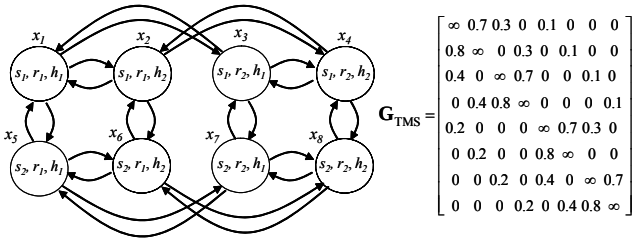
To construct the temperature state of the processor, we first define a set of temperatures $T_0 < T_1 < \ldots < T_c$, where $T_0 = T_{trig}$ (i.e., the trigger temperature threshold) and $T_c = T_{crit}$ (i.e., the critical temperature threshold). The intervening temperature thresholds are defined according to the ACPI (Advanced Configuration and Power Interface) specification. Once the temperature of the processor reaches the initial trigger temperature, the thermal manager is awakened to consider the conditions and issue a thermal management decision (i.e., a system state-changing command), ensuring that the critical temperature threshold is not exceeded. Thus, the CTMDP model of the chip temperature includes a set of temperature states $H = \{h_1, h_2, \ldots, h_c, h_{c+1}\}$ and a generator matrix $\mathbf{G}_{temp}$, where c+1 is the number of states that are possible with a thermal sensor. Note that h_i represents the temperature region between T_{i-1} and T_i, and h_{c+1} represents the temperature region that lies beyond T_{crit}. The transition rates in $\mathbf{G}_{temp}$ can be calculated as the inverse of the time it takes for the temperature of the processor to increase (decrease) from temperature state h_i to h_{i+1} (h_{i-1}), assuming that a thermal sensor receives streams of continuous-valued sensor data so that state h_i cannot evolve into state h_{i+2} or h_{i-2} directly.

3.3 Integrated Model of a TMS

After constructing the CTMDP models of the processor, application, and temperature reading, we denote by X the global state set of the integrated model, defined as the *Cartesian product* [18] of the state sets S, R, and H, with the generator matrix $\mathbf{G}_{TMS}$ which contains the transition rate from a global state $x = (s, r, h)$ to another $x' = (s', r', h')$. Note that the Cartesian product is a direct product of sets such as $S \times R \times H = \{(s, r, h) \mid s \in S, r \in R, \text{and } h \in H\}$. The global generator matrix $\mathbf{G}_{TMS}$ is calculated as the *tensor sum* [19] of generator matrices $\mathbf{G}_{proc}$, $\mathbf{G}_{app}$, and $\mathbf{G}_{temp}$. Note that when two CTMDPs with generator matrices $\mathbf{A}$ and $\mathbf{B}$ are given, the generator matrix of the joint process is obtained by the tensor sum, a matrix operator, of $\mathbf{A}$ and $\mathbf{B}$. Basically, the tensor sum, for example, $\mathbf{C} = \mathbf{A} \oplus \mathbf{B}$ is given by $\mathbf{C} = \mathbf{A} \otimes \mathbf{I}_{n2} + \mathbf{I}_{n1} \otimes \mathbf{B}$, where n_1 is the order of $\mathbf{A}$, n_2 is the order of $\mathbf{B}$, $\mathbf{I}_{ni}$ is the identity matrix of order n_i, and $\otimes$ is the *tensor product* [19]. The tensor product, for example, $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$, is defined as,

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} \end{bmatrix}, \quad if\ \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \tag{4}$$

where a_{11}, a_{12}, a_{21}, and a_{22} are scalars.



(a) (b)

Figure 5. An example of CTMDP model of a TMS (a) and its generator matrix (b).

An example of the integrated CTMDP model that captures temperature evolution is provided in Figure 5, assuming that we have two states for processor (s_1, s_2), application (r_1, r_2), and temperature (h_1, h_2), for simplicity. For example, if a micro-architectural configuration change from s_1 to s_2 takes place given application r_1 and temperature reading h_1, the TMS transits from x_1 to x_6 via x_5, where the temperature in the end evolves into h_2.

4. DYNAMIC THERMAL MANAGEMENT

In this section, mathematical formulation of the DTM problem that maximizes the performance metric subject to no exceeding a critical temperature threshold is constructed.

4.1 Optimal DTM Policy

After determining the relevant parameters for each state $x \in X$ and each arc in the CTMDP model of the TMS, we set up a mathematical programming model to solve the DTM problem as a linear program as below. The goal is to find an optimal state $s \in S$ which consists of action and micro-architectural configuration (a, c), while minimizing the energy cost of the system for a given level of performance and given an application r under tight temperature constraints. We call the tuple (a, c) a *command* since the TM controls the micro-architectural configuration and the VF setting, which in turn affect power dissipation of the processor, and thereby the resulting temperature.

$$\underset{s}{\text{mimimize}}\left(\sum_{x}\sum_{s_x} f_x^{s_x}\tau_x^{s_x}g_x^{s_x}\right) \tag{5}$$

s.t.
$$\sum_{s_x} f_x^{s_x} = \sum_{x'\neq x}\sum_{s_{x'}} f_x^{s_{x'}} p_{x',x}^{s_{x'}}\ , \quad \forall x \in X \tag{6}$$

$$\sum_{x}\sum_{s_x} f_x^{s_x}\tau_x^{s_x} = 1 \tag{7}$$

$$\sum_{x}\sum_{s_x} f_x^{s_x}\tau_x^{s_x}\delta(h(x), h_{c+1}) < \Pr_{crit} \tag{8}$$

where $f_x^{s_x}$ is the frequency that the system enters into state x with command s_x, $\tau_x^{s_x}$ is the expected duration of time that the system stays in state x when command s_x is taken, $g_x^{s_x}$ is the energy cost of the system for a given level of performance (i.e., the *energy-delay-squared product*, ED^{2}P, which captures the power-performance-efficiency under voltage scaling [8] and is independent of the clock frequency) when the system is in state x and command s_x is chosen, $p_{x',x}^{s_{x'}}$ is the probability that the next system state is x' if the system is currently in state x and command s_x is taken, $\delta(h(x), h_{c+1})$ is 1 if $h(x)$ (i.e., current h of state x) = h_{c+1} (i.e., temperature beyond T_{crit}) or 0 otherwise, and Pr_{crit} is a pre-defined threshold probability (i.e., the probability of exceeding the critical temperature threshold). $\tau_x^{s_x} = 1/\sum_{x'\neq x}\sigma_{x,x'}^{s_x}$.

The ED^{2}P metric may also be written as

$$ED^2P = \frac{Pwr}{Perf^3} \cong \frac{Pwr}{IPC^3} \tag{9}$$

where Pwr denotes the processor power consumption, and the processor performance is measured as the number of instructions per cycle (IPC). Pwr/IPC^3 is an excellent figure of merit to capture the energy cost of a given level of processor performance [8]. Note that we focus on AC line powered systems that strive to deliver maximum performance while operating under temperature constraints. Specifically, the purpose of this optimization problem is to maximize the system's power-performance-efficiency while constraining the probability that the peak temperature is greater than T_{crit} to be less than a pre-defined probability value, P_{crit}.

4.2 Online DTM

In many cases, we are unable to know the actual characteristics of the applications which are running on the processor in advance. Thus, we must also develop an online DTM technique by constructing a pre-characterized *configuration-command mapping* table, where the entries of this mapping table correspond to various combinations of application types and temperature readings. Figure 6 illustrates how the thermal manager interacts with the applications and the temperature readings. In this figure, the pre-characterized mapping table is obtained through extensive offline simulation during design time, considering every possible combination of states for processor, applications, and temperature readings. It is worthwhile to note that the thermal manager is initiated only when the temperature exceeds the initial trigger temperature threshold T_{trig} and then controls the performance of the processor by limiting critical temperature. More precisely, the thermal manager receives the states of current application and temperature when the temperature exceeds T_{trig}, and issues an optimal micro-architectural configuration and action set (i.e., command) to the processor.

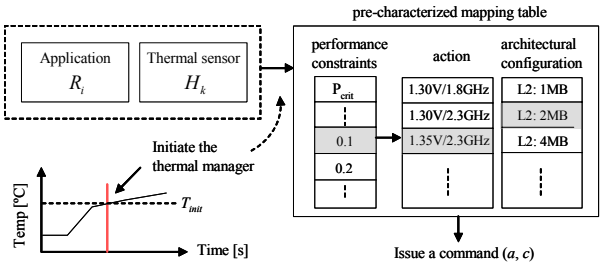


Figure 6. Online thermal management technique.

5. EXPERIMENTAL RESULTS

Experiments have been designed to evaluate the effectiveness of the proposed modeling technique and assess the performance of our optimization method. We use abstract models of the Intel Core Duo processor [20], which provides dynamic L2 cache resizing mechanism, to construct a TMS. Table 1 shows the transition time (normalized) for the CTMDP model of the processor, assuming that the system has $S = \{s_1, s_2, s_3, s_4\}$, where each state set $s = (a, c)$ is a combination of a_1 = [1.3V 1.8GHz], a_2 = [1.35V 2.3GHz], c_1 = [2MB L2 cache], and c_2 = [4MB L2 cache]. Note that due to the limitations of the simulation environment (e.g., *Vtune* performance analyzer [21]), we only consider a variable cache for the architectural configuration set (i.e., other configurable resources such as register file, reorder buffer, and load/store queue are not considered). Note that information about dynamic cache resizing time and voltage and frequency control lock time is obtained from [4][20].

Table 1. Transition times for the CTMDP model of the processor.

	(a_1, c_1)	(a_1, c_2)	(a_2, c_1)	(a_2, c_2)
(a_1, c_1)	0	1	0.8	1
(a_1, c_2)	5	0	5	0.8
(a_2, c_1)	0.8	1	0	1
(a_2, c_2)	5	0.8	5	0

To simplify the experimental setup, we consider $R = \{r_1, r_2, r_3, r_4\}$, where each r is a combination of two IPC ranges and two L2 cache miss rate (η) ranges: IPC ≤ 0.85, IPC > 0.85; $\eta \leq 0.01$, and $\eta > 0.01$, based on the performance distribution for application programs as shown in Figure 1. The initial trigger temperatures threshold is defined as T_{trig} = 60°C, with an ambient temperature of T_A = 40°C, based on the thermal design guideline, where we use the thermal performance data of a 35x35mm 478-pin micro-FCPGA package [20] to obtain temperature states. The on-chip temperature is estimated by utilizing $T_{chip} = T_A + P\cdot(\theta_{JA} - \Psi_{JT})$ based on the parameter values of the package. The device power dissipation P can be assumed to be a normally distributed random variable with some known mean value and standard deviation.

Figure 7 shows the results of the proposed DTM technique, where we randomly chose a sequence of 100 programs of SPEC CPU2000 (cf. Figure 1) with P_{crit} set to 0.2 and T_{crit} set to 71°C. An optimal architectural configuration and action set is selected and provided to the processor when an input (i.e., application and

Table 2. Power and performance comparisons between Greedy and SDTM techniques.

T_{crit}	Greedy		SDTM															
			P_{crit} = 0.05				P_{crit} = 0.15				P_{crit} = 0.25				P_{crit} = 0.35			
	Average Power	Average ED^{2}P	Average Power	Average ED^{2}P	Saving (%)		Average Power	Average ED^{2}P	Saving (%)		Average Power	Average ED^{2}P	Saving (%)		Average Power	Average ED^{2}P	Saving (%)	
					Power	Perf			Power	Perf			Power	Perf			Power	Perf
63°C	29.3	5.29	30.3	4.40	-3.0	16.8	31.9	4.31	-8.8	18.5	32.5	4.20	-10.9	20.6	32.7	4.15	-11.6	21.5
67°C	33.6	4.64	33.8	3.91	-0.6	15.7	33.9	3.90	-0.9	15.9	34.0	3.85	-1.2	17.0	34.2	3.82	-1.7	17.7
71°C	35.4	4.32	35.8	3.73	-1.1	13.6	36.2	3.70	-2.2	14.4	36.3	3.60	-2.5	16.7	36.5	3.52	-3.1	18.5
75°C	37.6	3.65	38.0	3.30	-1.0	9.6	38.3	3.25	-1.5	10.9	38.6	3.20	-2.6	12.3	38.9	3.01	-3.4	17.5

temperature state) is given to the mapping table, where the entries of this table correspond to various combinations of inputs and performance constraints. It is clearly seen that the peak power consumption, which results in the temperaure increase, is limited by constraining the probability that the peak temperature of the system is greater than T_{crit} to be less than P_{crit} in our DTM policy. The time steps are abstractly defined to represent the peak power value of each program run. As expected, constraining the power dissipation causes some performance (ED^{2}P) degradation. It, however, guarantees the thermal safety of the system.

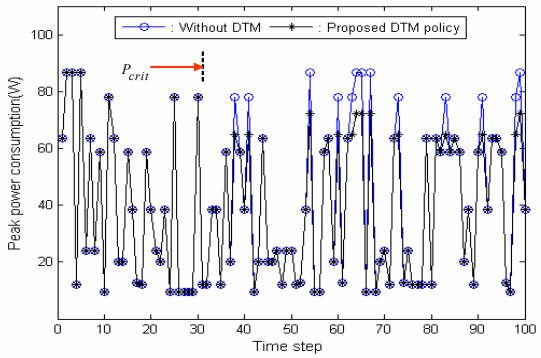


Figure 7. The effectiveness of the proposed DTM technique.

We investigated the performance-efficiency of the proposed DTM technique, which we call stochastic DTM, or SDTM for short. We assumed two voltage-frequency (VF) change commands are available (where $a_1 < a_2$ in terms of VF values). For comparison purpose, we also implemented a greedy DTM policy.

Greedy: Apply the following VF assignment strategy

- Use a_2 at low temperatures, i.e., $T_{trig} \leq T < (T_{trig} + T_{crit})/2$;
- Use a_1 at high temperatures, i.e., $(T_{trig} + T_{crit})/2 \leq T < T_{crit}$.

SDTM: Apply the optimal DTM commands, based on the mathematical program formulation of the TMS.

The Greedy policy gives considerable performance benefit, similar to clock throttling techniques which throttle clock and flush pipelines under temperature constraints. The simulation results in Table 2 (normalized), which varies the values of T_{crit}, demonstrate that, compared to the Greedy policy, the SDTM policy which allows exceeding T_{crit} to the degree of P_{crit} achieves performance savings of up to 16.1% (average) at the cost of 3.5% (average) power penalty. However, it indicates that as we move P_{crit} to smaller values (e.g., 0.05), we can achieve 13.9% (average) performance savings with little impact on the power metric.

6. CONCLUSION

We introduced a new technique for modeling and solving the DTM problem for multi-core systems. The proposed modeling technique, based on Markov decision processes captures dynamic characteristics of processor, applications, and die temperatures. From the mathematical model, we can calculate the optimal DTM policy, which maximizes the power-performance-efficiency under a peak temperature constraint.

7. REFERENCES

[1] W. Huang, M.R. Stan, K. Skadron, K. Sankaranarayana, S. Shosh, and S. Velusamy, "Compact Thermal Modeling for Temperature-Aware Design," *Proc. of DAC*, June, 2004.

[2] D. Brooks, and M. Martonosi, "Dynamic Thermal Management for High Performance Microprocessor," *Proc. of HPCA*, Jan., 2001.

[3] J. Srinivasan, and S. V. Adve, "Predictive Dynamic Thermal Management for Multimedia Applications," *Proc. of ACM Int'l Conference on Supercomputing*, Jun., 2003.

[4] A. Naveh, et al., "Power and Thermal Management in Intel Core Duo Processor," *Intel Tech. Journal*, Vol. 10, Issue 2, May, 2006.

[5] P. Dadvar, and K. Skadron, "Potential Thermal Security Risks," *Proc. of 21st IEEE Semi-Therm Symposium*, Mar., 2005.

[6] M. Janicki, and A. Napieralski, "Inverse Heat Conduction Problems in Electronics with Special Considerations of Analytical Analysis Methods," *Proc. of Int'l Semiconductor*, Vol. 2, Issue 4, Oct., 2004.

[7] M. L. Puterman, Markov *Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Publisher, New York, 1994.

[8] J. Silc, B. Robic, and T. Ungerer, *Processor Architecture: From Dataflow to Superscalar and Beyond*, Springer, 1999.

[9] D. C. Lee, et al., "Execution characteristics of desktop applications on Windows NT," *Proc. Int. Conf. on Computer Architecture*, 1998.

[10] CPU SPEC2000 documents. http://www.spec.org.

[11] S. Birdj, et al., "Performance Characterization of SPEC CPU Benchmarks on Intel's Core Microarchitecture based processor," *Proc. of 2007 SPEC Benchmark workshop*, Jan., 2007.

[12] JEDEC standards. http://www.jedec.org.

[13] E. Chung, L. Benini, and G. De Micheli, "Dynamic power management for non-stationary service requests," *Proc. of Design Automation and Test in Europe*, Apr., 1999.

[14] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, John Wiley & Sons, 2005.

[15] S. Lopez, et al., "Dynamic Capacity-Speed Tradeoffs in SMT Processor Caches," *Proc. of High Performance Embedded Architecture and Compiler*, Jan., 2007.

[16] D. Ponomarev, G. Kucuk, and K. Ghose, "Dynamic Resizing of Superscalar Datapath Components for Energy Efficiency," *IEEE Trans. on Computers*, Vol. 55, No. 2, Feb., 2006.

[17] A. Joshi, et al., "Measuring benchmark similarity using inherent program characteristics," *IEEE Trans. on Computers*, Vol. 55, No. 6, Jun., 2006.

[18] M. J. Osborne, *A Course in Game Theory*, MIT press, 1994.

[19] M. Davio, "Kronacker products and shuffle algebra," *IEEE Trans. on Computers*, Vol. 30, No.2, 1981.

[20] Intel Core Duo processor on 65nm process: Thermal Design Guide, Feb., 2006. http://www.intel.com.

[21] Vtune performance analyzer. http://www.intel.com/software.