Robust Control-theoretic Thermal Balancing for Server Clusters

Yong Fu*, Chenyang Lu*, Hongan Wang†
* Deptartment of Computer Science and Engineering, Washington University {fuy, lu}@cse.wustl.edu
† Institute of Software, Chinese Academy of Sciences wha@iel.iscas.ac.cn

***Abstract*—Thermal management is critical for clusters because of the increasing power consumption of modern processors, compact server architectures and growing server density in data centers. Thermal balancing mitigates hot spots in a cluster through dynamic load distribution among servers. This paper presents two *Control-theoretical Thermal Balancing (CTB)* algorithms that dynamically balance the temperatures of different servers based on online measurements. CTB features controllers rigorously designed based on optimal control theory and a difference equation model that approximates the thermal dynamics of clusters. Control analysis and simulation results demonstrate that CTB achieves robust thermal balancing under a wide range of uncertainties: (1) when different tasks incur different power consumptions on the CPUs, (2) when servers experience different ambient temperatures, and (3) when servers experience thermal faults.**

***Index Terms*—clusters; thermal balancing; feedback control;**

I. Introduction

Thermal management is important for server clusters due to increasing power consumption of modern processors, compact server architectures and growing server density in data centers. *Thermal balancing* aims to balance the temperatures of different processors through dynamic load distrbution in a server cluster. Thermal balancing is an attractive approach to thermal management in server clusters for three imporant reasons. First, thermal balancing can effectively mitigates hotspots in a cluster and hence effectively reduce the cooling cost for data centers. For example, a study of a real data center showed that reducing the temperature difference from $10°C$ to $2°C$ could result in close to a 25% reduction in total energy costs associated with the cooling infrastructure [1]. Second, in contrast to other thermal management mechanisms such as throttling and dynamic voltage scaling, thermal balancing can prevent server overheating without causing performance degradation. Finally, thermal balancing can be applied to heterogeneous server clusters including legacy processors that do not support throttling or dynamic voltage scaling.

To implement thermal balancing, a thermal balancer may be implemented on the gateway of a server cluster that provide a same set of services on multiple servers. The thermal balancer intercepts service requests from clients and then dynamically forwards them to appropriate servers based on online temperature measurement. While thermal balancing shares similarities with load balancing, it faces several unique challenges. First, while a load balancer is designed to balance the *load* on different servers, a thermal balancer aims to balance the *temperatures* of different servers. While the temperature of a server is related to its load, the thermal dynamics of a server are inherently more complex than system load, because the temperature not only depends on the current load but also its history. It is therefore important to incorporate the thermal dynamics in the design of the thermal balancing algorithm. Finally, thermal balancing must handle uncertain thermal characteristics, such as varying power consumption, thermal faults, and varying ambient temperature.

To tackle these challenges, we present *Control-theoretic Thermal Balancing (CTB)*, a novel thermal balancing approach based on a control-theoretic underpinning. CTB employs a feedback control loop that periodically monitors the temperature and CPU utilization of different servers in a cluster, and redistributes clients' service requests among different processors to dynamically balance their temperature. CTB features control algorithms that are rigorously designed and analyzed based on optimal control theory. Specifically, this paper makes the following main contributions.

- We derive a difference equation model that characterizes the thermal dynamics of server clusters as a foundation for control designs and analysis of thermal balancing.
- We present the design and stability analysis of two CTB algorithms analytically designed based on optimal control theory. CTB-T uses processor temperature as feedback while CTB-UT uses both processor temperature and CPU utilization to significantly reduce task reallocation cost.
- We provide simulation results that demonstrate CTB algorithms can deliver robust thermal balancing in face of a wide range of uncertainties, including different power consumption incurred by different tasks, different ambient temperatures of different servers, and thermal faults.

In the rest of this paper, Section II formulates the thermal balancing problem from a control perspective. Section III details the design of CTB algorithms. Section IV provides the simulation results. Section V introduces related works. Section VI concludes the paper.

II. Problem Formulation

In this section we first describe the system and thermal models and then formulate the thermal balancing problem.

A. System model

A cluster consists of n homogeneous single-processor servers $\{S_i | 1 \leq i \leq n\}$ connected by networks. All servers host a same set of services. A client may periodically invoke a service hosted by a server, where the periodic processing of the request corresponds to a periodic task T_i on the server. Let the execution time and the period of T_i are c_i and p_i, respectively. Then the (CPU) utilization of T_i is $U_i = \frac{c_i}{p_i}$.

As managing the temperature of processors is a major concern in server cluster, we focus on the thermal and power properties of processors. Extending our work to thermal control for the other system components is part of our future work. The processor of each server has *estimated* active power P_a (e.g., the active power in the specification of the processor) when it is executing tasks. It is important to note that the *actual* active power of a processor may deviate from the estimated at run time and different tasks may incur different power consumption [2], [3]. For example, an earlier study showed that the active power consumption of different applications may differ by as much as 35% [2]. In earlier literature some researchers [4] referred to such significant power variation during run time as *power phase* behavior. At the instruction level, different instruction types, inter-instruction overhead, memory system state and pipeline related effects cause fluctuation of task powers [5]. When the processor is idle, the processor switches to a low power mode and consumes power of P_{idle}.

We employ a widely adopted thermal model [6]–[8] for the processor Pr_i as follows.

$$\frac{dT_i(t)}{dt} = -c_{i,2}(T_i(t) - T_0) + c_{i,1}P_i(t) \tag{1}$$

where $T_i'(t)$ is the temperature of processor Pr_i, $c_{i,1}, c_{i,2}$ are the constant pertained to the thermal characteristics of the processor and T_0 is ambient temperature. Let $T_i'(t) = T_i(t) - T_0$. We can write equation (1) in a more compact form

$$\frac{dT_i'(t)}{dt} = -c_{i,2}T_i'(t) + c_{i,1}P_i(t).$$

For the whole system with n homogeneous processors, the thermal model is the aggregation of the individual processor's thermal model,

$$\dot{\mathbf{T}}'(t) = \mathbf{A}\mathbf{T}'(t) + \mathbf{B}\mathbf{P}(t)$$

where

$$\mathbf{T}'(t) = [T_1'(t), T_2'(t), ..., T_n'(t)]^T$$

and

$$\mathbf{P}(t) = [P_1(t), P_2(t), ..., P_n(t)]^T$$

while the constant matrices are

$$\mathbf{A}=\begin{bmatrix} -c_{1,2} & & 0 \\ & \ddots & \\ 0 & & -c_{n,2} \end{bmatrix}, \mathbf{B}=\begin{bmatrix} c_{1,1} & & 0 \\ & \ddots & \\ 0 & & c_{n,1} \end{bmatrix}.$$

Thus, the thermal model of the cluster can be written as

$$\frac{d\mathbf{T}'(t)}{dt} = \mathbf{A}\mathbf{T}'(t) + \mathbf{B}\mathbf{P}(t) \tag{2}$$

Our thermal model does not consider the correlation of the temperatures of different servers based on recent studies that showed the thermal correlation between severs is insignificant in a commercial server cluster [9].

B. Dynamic Model for Thermal Balancing

As our CTB algorithms are designed to control the processors' temperatures through load redistribution, we need to establish a difference equation model to characterize their dynamic relationship between the processors' temperature and their CPU utilization, where the CPU utilization is the fraction of time when the CPU is running tasks. ufor the purpose of control design. As the feedback control loops of CTB are invoked once every sampling period W, we first discretize the continuous thermal model. Let $P(k)$ and $T'(k)$ denote the discretized power and temperature, respectively, which are measured at sampling time kW.

By *bilinear transformation* [10], the continuous form of the thermal model (2) is discretized as

$$\mathbf{T}'(k+1) = \mathbf{\Phi}\mathbf{T}'(k) + \mathbf{\Gamma}'\mathbf{P}(k) \tag{3}$$

where $\mathbf{\Phi} = (\mathbf{I} + \frac{\mathbf{AW}}{\mathbf{2}})(\mathbf{I} - \frac{\mathbf{AW}}{\mathbf{2}})^{-1}$ and $\mathbf{\Gamma}' = (\mathbf{I} - \frac{\mathbf{AW}}{\mathbf{2}})^{-\mathbf{1}}\mathbf{B}\sqrt{W}$ and W is the length of sampling interval.

Next we characterize the relationship between the CPU utilization and the power consumption of a processor. Let $U_i(k)$ denote the CPU utilization of the processor Pr_i in the k^{th} sampling period. Then the average power of the processor in k^{th} sampling period, $\bar{P}_i(k)$, can be written as

$$\begin{aligned} \bar{P}_i(k) &= G_i P_a U_i(k) + P_{idle}(1 - U_i(k)) \\ &= G_i(P_a - P_{idle})U_i(k) + P_{idle}. \end{aligned} \tag{4}$$

where G_i is the ratio between the average *actual* and *estimated* active power of the processor. Note that G_i is unknown at design time. An important goal of the our work is to design a control algorithm for thermal balancing that can tolerate a wide range of variations in G_i.

For the thermal control analysis we need to derive a discrete-time model to approximate this system. As the thermal-time constant is large, the effects of transients of power consumption within a sampling period is negligible. Therefore, we substitute average power $\bar{\mathbf{P}}(k) = [\bar{P}_1, \bar{P}_2, ..., \bar{P}_n]^T$ with discrete power $\mathbf{P}(k)$ in (3). Combining (3) and (4) and considering all processors in the server cluster, we get the following dynamic model for the server cluster:

$$\mathbf{T}'(k+1) = \mathbf{\Phi}\mathbf{T}'(k) + \mathbf{\Gamma}'\mathbf{G}(\mathbf{P}_a - \mathbf{P}_{idle})\mathbf{U}(k) + \mathbf{\Gamma}'\mathbf{P}_{idle} \tag{5}$$

where $\mathbf{G}$ is defined as $diag(G_1, G_2, ..., G_n)$, $\mathbf{P}_a = [P_a, P_a, ..., P_a]^T$ and $\mathbf{P}_{idle} = [P_{idle}, P_{idle}, ..., P_{idle}]^T$. Let $\mathbf{T}(k) = \mathbf{T}'(k) - \tilde{\mathbf{T}}$, where $\tilde{\mathbf{T}} = \mathbf{\Gamma}'\mathbf{P}_{idle}(\mathbf{\Phi} - \mathbf{I})^{-1}$, the thermal model of the system can be rewritten as

$$\mathbf{T}(k+1) = \mathbf{\Phi}\mathbf{T}(k) + \mathbf{\Gamma}\mathbf{U}(k) \tag{6}$$

where $\mathbf{\Gamma} = \mathbf{G}(\mathbf{P_a} - \mathbf{P_{idle}})\mathbf{\Gamma}'$.

It is noted that our model ignores the discrete nature of task utilizations and thus more suitable for servers with a large number tasks each consuming a small fraction of the CPU cycles. This liquid model is a reasonable approximation of many high-performance servers. Extending our work to deal with servers with non-negligible discrete utilization changes is part of our future work.

C. Thermal Balancing Objective

The objective of thermal balancing is,

$$\min_{k\to\infty} \sum_{i=1}^{n} (T_i(k) - \bar{T}(k))^2 + \rho \sum_{i=1}^{n} \Delta U_i^2(k) \qquad (7)$$

where $\bar{T}(k)$ is the average temperature, defined as $\bar{T}(k) = \frac{\sum_{i=1}^{n} T_i(k)}{n}$, $\Delta U_i(k)$ is the change to the utilization of processor Pr_i, i.e., the difference between the total utilization of the tasks moved to processor Pr_i and that of the tasks moved from processor Pr_i.

The first term of the objective function aims to reduce the differences among the temperatures of different processors. The second term of the objective function aims at reducing control cost, *i.e.*, the number of tasks redirected. This is important because redirecting a tasks can incur non-negligible performance penalty and overhead , e.g., due to loss of cache states or reestablishing the HTTP session.

III. CTB Design and Analysis

In this section we first provide an overview of the *Control-theoretic Thermal Balancing (CTB)* approach. We then present the difference equation model that characterizes the thermal dynamics of a servers cluster. Based on the dynamic model we detail the design and stability analysis of two CTB algorithms.

A. Overview of CTB

CTB uses online feedback for thermal balancing. A natural choice of feedback is temperature (the controlled variable). However, as temperature responds slowly to load redistribution, thermal balancing solely based on temperature feedback may not be able to regulate temperature quickly. To deal with the problem, the thermal balancer may also employ CPU utilization as feedback. In this work we develop two control algorithms for thermal balancing. *CTB-T* only uses the temperature as feedback while *CTB-UT* employs both utilization and temperature as feedback.

As shown in Figure 1, CTB employs a distributed feedback control loop consisting of a *controller* and a *balancer* on the gateway for the cluster, and *monitors* located on the servers. For the CTB-T algorithm, the controller input is a vector, $\mathbf{T(k)}$, which includes the temperature of each processor at the end of the kth sampling period. The output of the controller is a vector of *utilization change*, $\mathbf{\Delta U(k)}$, which indicates the requested change to each processor's utilization. According to the output of the controller, the *balancer* computes the clients' service requests invocations that should be redirected among different servers in the following sample period.

Specifically, CTB-T algorithm works as follows. At the end of the kth sampling period, the feedback loop is invoked and executes the following steps:

1) Each temperature *monitor* sends the temperature in the end of the last sampling period to the controller. Most modern processors integrates on-chip temperature sensors. Alternatively, software techniques based on event counters can be used to estimate processor temperature [8].
2) The *controller* calculates the change to the CPU utilization of every processor, $\mathbf{\Delta U(k)}$, based on the temperature vector $\mathbf{T(k)}$. Then $\mathbf{\Delta U}(k)$ is sent to centralized *balancer*.
3) The *balancer* reallocates tasks among different servers to accommodate the requested utilization change $\mathbf{\Delta U}(k)$. The Balancer first divides all processors into three sets, the *receivers*, the *senders*, and the *neutral* according to the controller output $\Delta U(k)$. The processors in the receivers set have positive utilization changes; the processors in the senders set have negative utilization changes; and the processors in the neutral set have zero utilization change and hence are not involved in task reallocation in the following sampling period. The Balancer then reallocate tasks from processors in the senders set to processors in the receivers set to accommodate the requested utilization changes specified in $\Delta U(k)$. In the following sampling period, the Balancer directs clients' service invocations to appropriate servers according to the new task allocation.

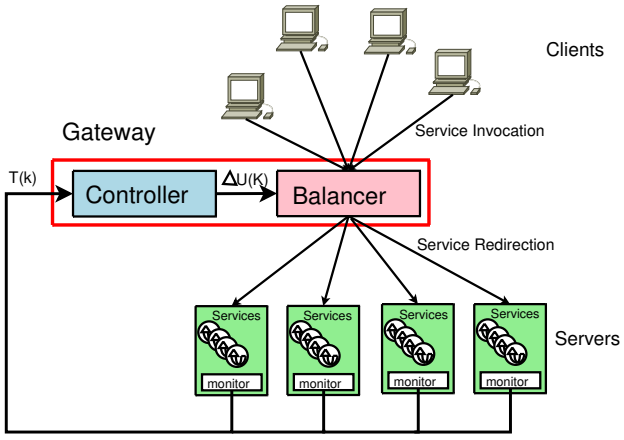


Fig. 1. The Feedback Control Loop of CTB-T

CTB-UT works in the same way as CTB-T except that it employs a different control algorithm that utilizes both the CPU utilization vector $\mathbf{U}(k)$ and the temperature vector $\mathbf{T}(k)$ as control inputs. Each processor hence runs a utilization monitor in addition to the temperature monitor. The utilization monitor measures the CPU utilization in the last sampling period and sends it to the controller. For example, in Linux, the utilization monitor can use `/proc/stat`file to estimate the CPU utilization in each sampling period. The `/proc/stat`file records the number of *jiffies* since the

system start time, when the CPU is in user mode, user mode with low priority (nice), system mode, and when used by the idle tasks. At the end of each sampling period, the utilization monitor reads the counters, and estimates CPU utilization by dividing the number of *jiffies* used by the idle tasks in the last sampling period by the total number of *jiffies* in the same period [11].

B. Control Design of CTB-T

The core of the CTB algrithm is the controller. Recall the control design has two objectives: (1) to reduce the differences among the temperature of different processors, and (2) to reduce the number of reallocated tasks to reduce overhead. The first goal aims at thermal balancing, while the second goal aims at reducing the control cost. To address both objectives, we choose to design a Linear Quadratic Regulator (LQR) based on optimal control theory. The LQR controller is designed for the following optimization objective.

$$\min_{k\to\infty}\sum_{k}[\mathbf{X}(k)^T\mathbf{Q}\mathbf{X}(k)] + \Delta\rho\mathbf{U}(k)^T\mathbf{\Delta U}(k)]. \tag{8}$$

where $\mathbf{Q}$ are weight matrix respectively and $\mathbf{\Delta X}(k)$ is the state of the thermal model (6), *i.e.*, $\mathbf{X}(k) = \mathbf{T}(k)$. If we denote

$$\mathbf{L} = \begin{bmatrix} 1-\frac{1}{n} & -\frac{1}{n} & \dots & -\frac{1}{n} \\ -\frac{1}{n} & 1-\frac{1}{n} & \dots & -\frac{1}{n} \\ \vdots & \ddots & \dots & -\frac{1}{n} \\ -\frac{1}{n} & \dots & \dots & 1-\frac{1}{n} \end{bmatrix}$$

and then $\mathbf{Q} = \mathbf{L^T L}$, the thermal balancing objective (7) can be transformed to the LQR controller with objectives 8. In fact because the difference between processor Pr_i's temperature and the average temperature of all processors is

$$\begin{aligned} \Delta T_i(t) &= T_i(t) - \frac{\sum_{1<k<n} T_k(t)}{n} \\ &= (1-\frac{1}{n})T_i(t) - \frac{1}{n}T_1(t)\dots - \frac{1}{n}T_n(t) \end{aligned} \tag{9}$$

The optimization objectives of our LQR controller 8 clearly match the thermal balancing and the control cost objectives 7.

C. Control Design of CTB-UT

There is a significant delay between the change in the CPU utilization and the resultant change in the processor temperature due to slow thermal dynamics. To improve the responsiveness of thermal balancer, CTB-UT employs both temperature and utilization as feedback for thermal balancing. CTB-UT also employs an LQR controller, but its state variables include temperatures and utilization.

The dynamics of utilization can be modeled as [12]

$$\mathbf{U}(k+1) = \mathbf{U}(k) + \mathbf{\Delta U}(k). \tag{10}$$

Combining 10 and thermal model (6), we can model the relationship between the temperatures and the utilization

$$\begin{aligned} \begin{bmatrix} \mathbf{T}(k+1) \\ \mathbf{U}(k+1) \end{bmatrix} &= \begin{bmatrix} \Phi & \Gamma \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{T}(k) \\ \mathbf{U}(k) \end{bmatrix} \\ &+ \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \mathbf{\Delta U}(k) \\ &= \mathbf{\Phi_{UT}} \begin{bmatrix} \mathbf{T}(k) \\ \mathbf{U}(k) \end{bmatrix} + \mathbf{\Gamma_{UT}}\mathbf{\Delta U}(k) \end{aligned} \tag{11}$$

The LQR controller of CTB-UT has the form as follows

$$\mathbf{\Delta U}(k) = -\begin{bmatrix} \mathbf{K_T} & \mathbf{K_U} \end{bmatrix} \begin{bmatrix} \mathbf{T}(k) \\ \mathbf{U}(k) \end{bmatrix} \tag{12}$$

The optimization objective of the LQR controller of CTB-UT is:

$$\begin{aligned} \min_{k\to\infty}\sum_{k}\Bigg\{ &\begin{bmatrix} \mathbf{T}(k) \\ \mathbf{U}(k) \end{bmatrix}^T \mathbf{Q_{UT}} \begin{bmatrix} \mathbf{T}(k) \\ \mathbf{U}(k) \end{bmatrix} \\ &+\mathbf{\Delta U}^T(k)\mathbf{R_{UT}}\mathbf{\Delta U}(k)\Bigg\} \end{aligned} \tag{13}$$

where $\mathbf{Q_{UT}} = \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{0} & \lambda \end{bmatrix}^T \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{0} & \lambda \end{bmatrix}$, $\mathbf{R_{UT}} = \rho\mathbf{I}$ and λ is a tunable parameter. The first term of optimization objective is responsible for balancing both temperature and utilization. Through tuning λ, we can adjust the weight between temperature balancing and utilization balancing. If $\lambda < 1$, the controller is biased for thermal balancing. If $\lambda > 1$ the controller is biased for load balancing which responds quickly to load change. With an appropriate λ, CTB-UT can combine the benefits of thermal balancing and load balancing.

D. Stability and Robustness

The stability of a server cluster with CBT is defined as convergence of the temperatures of all processors to their average temperatures. We design the LQR controllers based on a *nominal* system with $G = I$, that is, the power and utilization equals their estimation. It is important to derive the region of G where the system remains stable.

Theorem 1: The stable range of the $\mathbf{G}$ in CTB-T is

$$diag\left(\frac{1}{1+\sqrt{\alpha}}\right) < \mathbf{G} < diag\left(\frac{1}{1-\sqrt{\alpha}}\right), \tag{14}$$

where

$$\alpha = \frac{R}{R + W\Gamma^T S\Gamma}.$$

and W is the sampling period and S is the solution of Algebraic Riccati Equation (ARE) [10], $S = \Phi^T[S - S\Gamma R^{-1}\Gamma^T S]\Phi + Q$.

The proof of Theorem 1 can be derived directly from Corollary 11.4.1 in [13]. It is noted that since $R = \rho I$ the robust stable region of CTB-T varies with ρ.

Note this approach to robustness analysis is not applicable to CTB-UT. When power gain G varies, Φ_{UT} also changes. Thus we cannot use Theorem 1 to derive the stability region since the ARE does not have a fixed solution like the case of

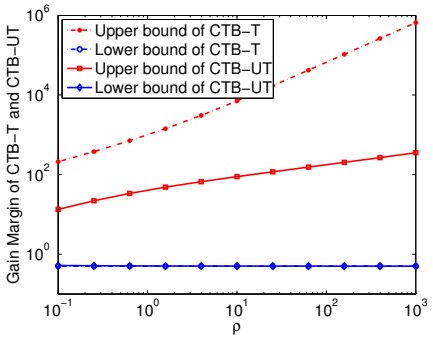


Fig. 2. Analytical Power Gain Stable Region of CTB-T and CTB-UT

CTB-T. Instead, we develop a numerical solution to calculate the stability region.

First we derive the closed-loop systems based on thermal model of CTB-UT (11) and the optimal controller (12). The closed-loop system has the form

$$\begin{aligned}\begin{bmatrix} T(k+1) \\ U(k+1) \end{bmatrix} &= (\Phi_{UT} - \begin{bmatrix} K_T & K_U \end{bmatrix} \Gamma_{UT}) \begin{bmatrix} T(k) \\ U(k) \end{bmatrix} \\ &= \Phi_C \begin{bmatrix} T(k) \\ U(k) \end{bmatrix}\end{aligned} \tag{15}$$

For a fixed ρ, Φ_C characterizes dynamics of the closed-loop system. According to control theory, if the eigenvalues of Φ_C is in the unit circle the dynamical system is stable. We can acquire the stable region of G based on the eigenvalues of Φ_C under different values of G.

Figure 2 illustrates the stable region of CTB-T and CTB-UT derived analytically. Both CTB algorithms can maintain stability under a wide range of power gains G. For example, when $\rho = 1$ the stable gain margin range of CTB-T and CTB-UT are $[0.50, 1474.30]$ and $[0.51, 42.91]$ respectively. CTB-UT has a smaller stability region than CTB-T. Therefore the choice between the them should consider the tradeoff between their responsiveness to variations and their robustness with regard to varying power consumptions. Another trends is that the stable region of both CTB-T and CTB-UT increases with ρ, the relative weight of the control cost and the temperature difference in the optimization objective. A higher ρ, leads to lower control cost and hence a higher degree of robustness against high power consumption.

IV. Evaluation

We evaluate our CTB algorithms using simulations. We first describe the simulation setup and three baseline algorithms for comparison. We then presents simulation results with varying power consumption, thermal faults, and varying ambient temperatures.

A. Simulation Setup

To evaluate the CTB algorithms we develop an event-driven simulator which simulates a cluster consisting of 16 servers. The CTB algorithms are implemented using the Optimal Control Toolbox of MATLAB.

The task set running on each processor consists of 50 periodic soft real-time tasks. The Eearlies Deadline First (EDF) scheduling algorithm [14] is employed to schedule all these tasks. The period p_i of each task T_i is chosen randomly with a uniform distribution in the range $[900ms, 1100ms]$. The deadline of each task equals its period.

Each processor simulated is a $2.6GHz$ Pentium 4 (P4) processor with $130nm$ Northwood core. All thermal related parameters except thermal capacitance shown in Table I are based on the Intel technical specification [15]. The thermal capacitance is acquired by simulating P4 on Hotspot [7], an architecture level simulator.

TABLE I
Power and Thermal Parameters

Parameter	Notation	Value
Ambient temperature	T_0	$45°C$
Max case temperature	T_c	$75°C$
Estimated active power	P_a	$51.9W$
Idle power*	P_i	$13.3W$
Thermal capacitance	C_{th}	$295.7J/K$
Thermal resistance	R_{th}	$0.467K/W$

* Enhanced Halt Mode is available [16]

B. Baseline Algorithms

We use three baseline algorithms as baselines for comparison in our simulations: an OPEN-loop algorithm (OPEN), a Load Balancing (LB) algorithm and a Heuristic Thermal Balancing (HTB) algorithm. OPEN does not perform any thermal or load balancing. Tasks are always executed on their initial processors.

The LB algorithm is designed to dynamically balance the CPU utilization of different processors. The invocation period of the LB algorithm is the same as the sampling period of CTB. In the end of each period, LB measures the utilization of every processor in the last sampling period and then redistributes the tasks to balance the utilization of different processors.

Like the CTB algorithms, the HTB algorithm is designed to balance the temperatures of different processors based on temperature feedback. In contrast to the optimal control approach adopted by CTB, HTB employs a simple heuristic algorithm to reallocate tasks among the processors. The heuristics is based on the observation that the steady-state temperature is proportional to the utilization of the processor. In the end of each sampling period, HTB changes the utilization of a processor proportionally to the difference between its temperature and the average temperature of all processors, i.e., $\delta u_i = f_t * [\frac{T_1}{n} + ...(1 - \frac{T_i}{n})... + \frac{T_n}{n}]$. f_t is the ratio between the steady temperature and the utilization, deriving by setting that $\mathbf{T}(k+1) = \mathbf{T}(k)$ in thermal dynamic equation (6). Note that the HTB algorithm has two key differences from the CTB algorithms: (1) it is not designed to reduce the number of tasks redistribution (*i.e.*, control cost); and (2) it ignores the thermal dynamics of the system which may influence the transient response to system variations.

C. Effect of Thermal Balancing

The first set of simulations evaluates the capability of the CTB algorithms to achieve thermal balance. Each simulation run lasts for 3000s. At starting time, the utilization of each processor is assigned randomly in the range $[0.5, 0.7]$. The power ratio of tasks are randomly selected in the range $[0.6, 1]$. We set the tunable parameter $\rho = 1.0$ for CTB, and $\rho = 1, \lambda = 0.005$ for CTB-UT.

The comparison between CTB and the baseline algorithms are illustrated in Figure 3. Since no temperature balancing is applied under OPEN, the maximum temperature difference between processors is $11.8°C$ as shown in Figure 3(a). While LB effectively balances the utilization of the processors, the maximum temperature difference is reduced only slightly to $9.9°C$. This results shows that load balancing is not effective in balancing temperature in server clusters due to the varying power consumption among different tasks. In contrast, the feedback-based thermal balancing algorithms, CTB-T, CTB-UT and HTB, effectively balance the temperatures reducing the temperature difference within $9.9°C$ in steady states.

The comparison between LB and the thermal balancing algorithms indicates inherent tradeoff between performance and temperature in server clusters. LB results in more balanced CPU utilization among processors which may lead to higher average performance. The thermal balancing algorithms, on the other hand, results in more balanced temperature at the cost of different CPU utilization among servers. Thermal management has become increasingly important due to the extremely high cooling cost in data centers today [1]. Note the reduction of the maximum temperature difference from $9.9°C$ (under LB) to $0.2°C$ (under CTB algorithms) can have a significant impact on the cooling cost of server clusters. For example, previous work showed that reducing the temperature difference from $10.0°C$ to $2°C$ will result in close to a 25% reduction in total energy costs associated with the cooling infrastructure [1].

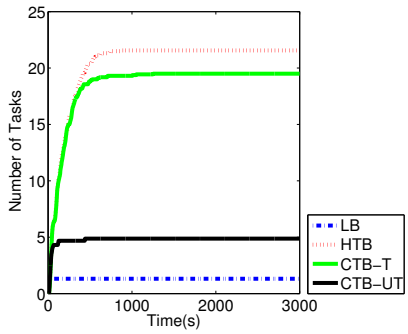
D. Comparison of Thermal Balancing Algorithms

While HTB and both CTB algorithms balance temperatures effectively, there are two important differences in their performance. First, the CTB algorithms, particularly CTB-UT, converges to a steady state with balanced temperature significantly faster than HTB. Table II compares the *convergence time*, *i.e.*, the time from beginning of the run to the time instant when the last task reallocation is completed. The convergence time of CTB-UT is only one quarter of CTB-T and one sixth of HTB. This result shows that CTB-UT is the most responsive to system variation. As discussed earlier, the responsiveness of CTB-UT results from its design that uses utilization in addition to temperature as feedback.

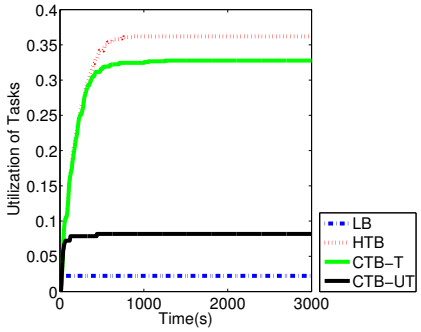
Second, the overhead of CTB-UT is also lower than HTB and CTB-T. As shown in Figure 4, on average HTB and CTB-T reallocated 7.8 and 11.0 tasks, respectively, per processor before converging to steady states. In comparison, CTB-UT converges to the steady state after reallocating only 5 tasks per processor. This result demonstrates the effectiveness of incorporating control cost in the optimal control design, especially when combined with both utilization and temperature as feedbacks.

Algorithm Type	OPEN	LB	HTB	CTB-T	CTB-UT
Converge Time(s)	N/A	10	1540	870	260
Max. Temp. Diff.($°C$)	11.9	9.9	0.2	0.2	0.2

TABLE II
COMPARISON OF DIFFERENT ALGORITHMS



(a) Reallocated Tasks per Processor

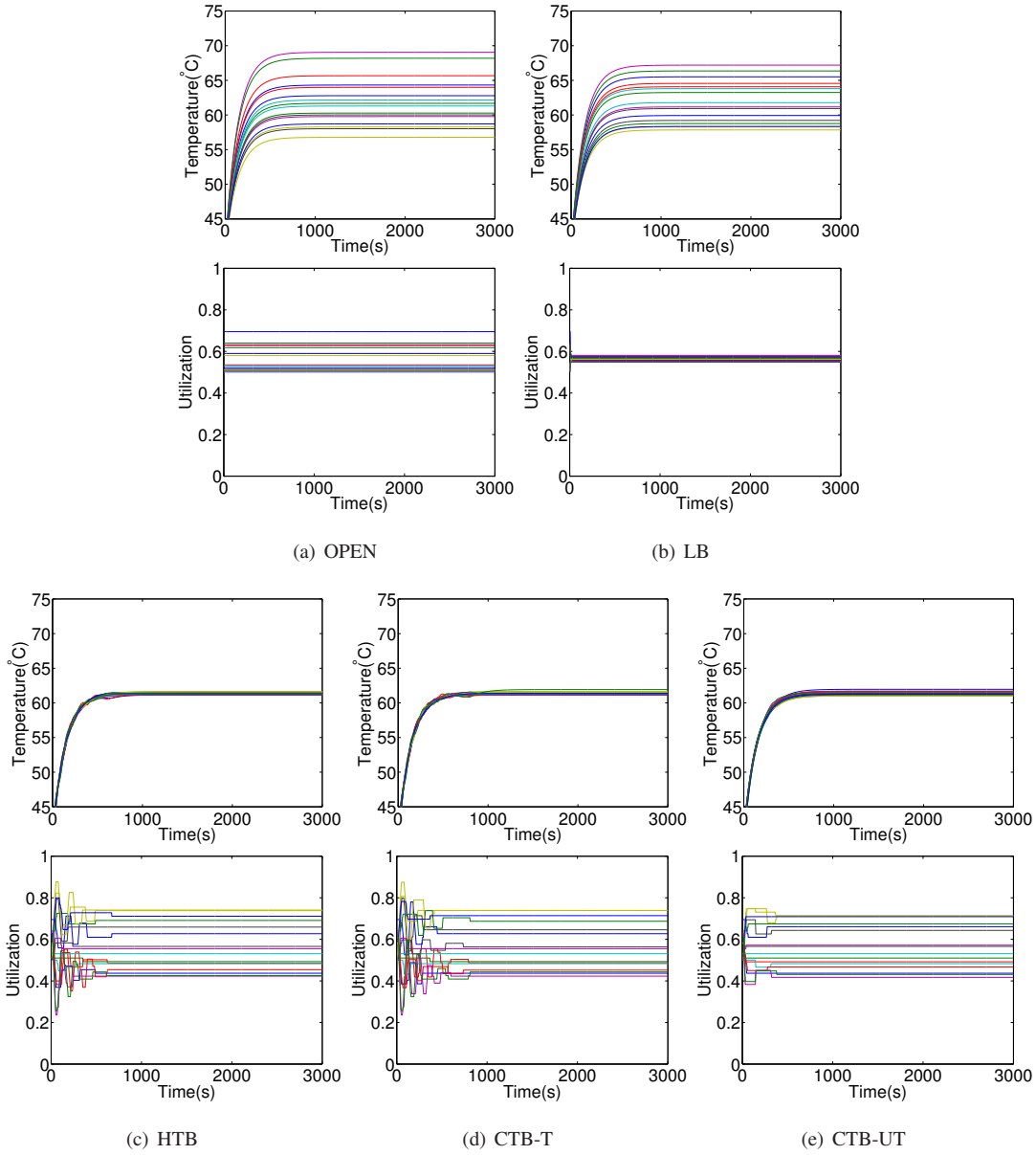


(b) Average Total Utilization of Reallocated Tasks per Processor

Fig. 4. Comparison of Overhead due to Tasks Reallocation

To compare the algorithms with a wide range of power consumption, we rerun the simulation with different power distributions for the task set. As shown in Figure 5, all the thermal balancing algorithms (HTB, CTB-T, and CTB-UT) effectively maintains temperature balance under the wide range of power ratios used in the simulations, while LB and OPEN result in significant differences in temperatures. Moreover, CTB-UT consistently outperforms CTB-T and HTB in term of reallocation cost.

Figure 5 shows the temperature difference and overhead of different algorithms when power distribution of tasks changed. The horizontal axis of the figure is the lower bound of power range, for example, 0.8 means the power ratio of tasks distributes in the range $[0.8, 1]$. In all power distribution, CTB achieves the equivalent temperature difference of HTB and has significant overhead reduction.



(a) OPEN (b) LB

(c) HTB (d) CTB-T (e) CTB-UT

Fig. 3. Temperatures and CPU utilization of all processors under CTB and baseline algorithms

E. Thermal Fault

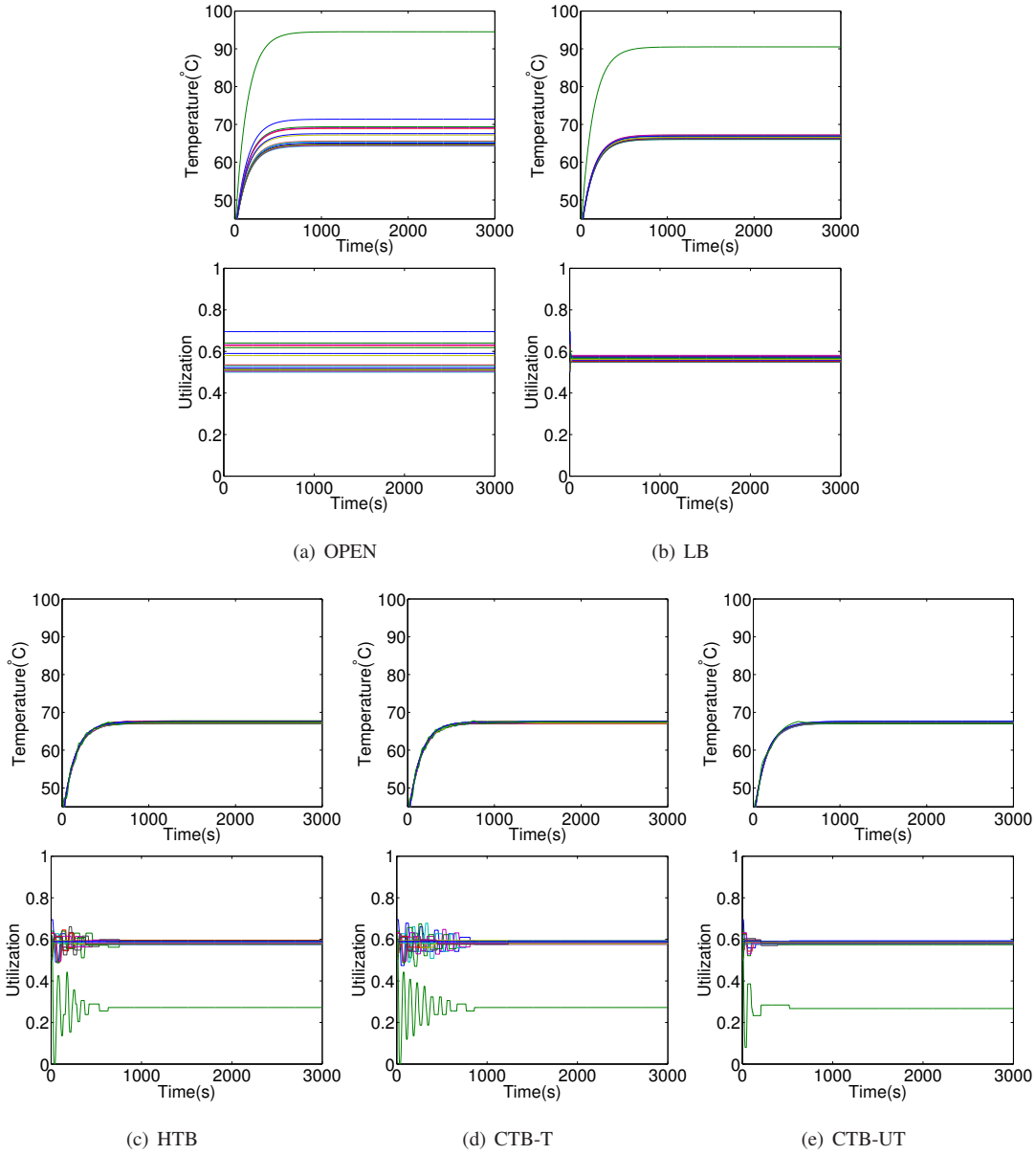
This set of simulations test the system's capability to handle thermal faults in servers such as the failure of their cooling system. In this case a robust thermal balancing algorithm should dynamically reallocate tasks from the servers with thermal faults to other servers to maintain thermal balance in the cluster when possible. We simulate the failure of a fan in a server by doubling its thermal resistance, R_{th} [17], in the beginning of the run.

Figure 6 shows the simulation results. As expected, OPEN and LB cannot deal with the temperature increase in the server with thermal fault. In contrast, the thermal balancing algorithms effectively maintain the thermal balance by reducing the utilization of the server with thermal fault. This result demonstrates the robustness of the feedback-based thermal balancing algorithms. As shown in Figures 7, CTB-UT induces significantly lower cost than CTB-T and HTB for task reallocation.

F. Robustness against Different Ambient Temperatures

We now evaluate the algorithms' capability to handle the case where different processors experience varying ambient temperatures. In each run the difference of ambient temperature of processors is distributed uniformly in the rang $[40°C, 50°C]$

As shown in Figure 8, all thermal balancing algorithms effectively balances the temperatures despite varying ambient temperatures among processors, while LB results in significant differences in processor temperatures. Furthermore, CTB-UT



(a) OPEN (b) LB

(c) HTB (d) CTB-T (e) CTB-UT

Fig. 6. Temperatures and CPU utilization of all processors under CTB and baseline algorithms

incurs the lowest cost in term of task reallocations among the thermal balancing algorithms as shown in Figure 9.

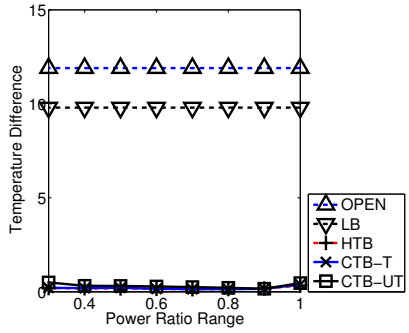
To further test the robustness of the algorithms against variations in ambient temperatures, we repeat the simulations with ambient temperatures varying in different ranges. As shown in Figure 10, all thermal balancing algorithms maintain thermal balance even when the ambient temperatures od different processors vary by as much as $20°C$. CTB-UT consistently leads to significantly fewer task reallocations than CTB-T and HTB. These results show the robustness and efficiency of CTB-UT under varying ambient temperatures.

V. Related Works

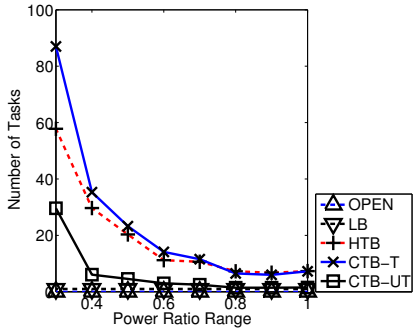
Feedback-based thermal management has been applied at different levels of computer systems. At the architecture level, the authors of [6] employ feedback control to regulate the instruction fetch rate to control the temperature of the processor. Feedback control is also used to manipulate clock gating for mitigating the thermal pressure [2]. Heat-and-Run [18] balances the temperature of different cores in a multicore processor through instruction migration.

At the single-node system level, a predictive dynamic thermal management method [19] has been proposed to regulate the temperature generated by multimedia applications. At the distributed system level, Weatherman [20] adopts a *data-driven* method to derive the heat distribution in data centers and then use this distribution to adjust workload in the data centers.

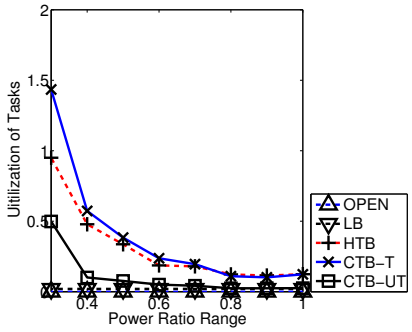
The work most closed related to this paper is Power Balancing [21]. It used an event counter to estimate the power of tasks



(a) Temperature Difference

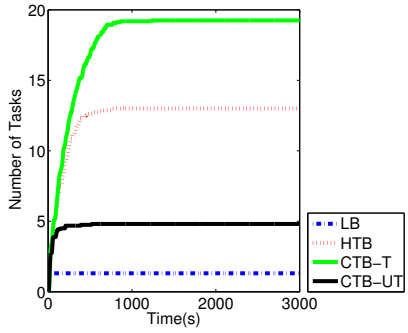


(b) Average Number of Reallocated Tasks

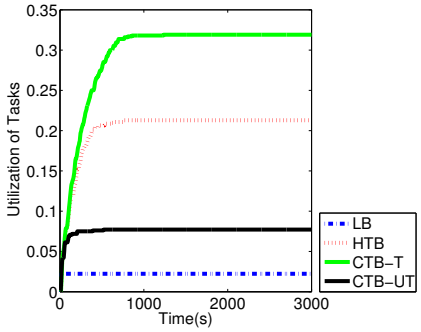


(c) Average Utilization of Reallocated Tasks

Fig. 5. Comparison of Algorithms with Different Range of Power Ratio. The x axis represents the lower bound of the power ratio. For each data point shown in this figure, the power ratio of tasks are randomly chosen in the range $[x, 1]$.



(a) Reallocated Tasks per Processor



(b) Average Total Utilization of Reallocated Tasks per Processor

Fig. 7. Comparison of Overhead due to Tasks Reallocation.

on at run time and then performs thermal balancing through task migration in a multi-processor system. However, this work performs thermal balancing based on power estimation instead of direct temperature measurement. Thus different applications need different parameters that need to be calibrated. Furthermore, their algorithms are not analytically designed based on a control-theoretic approach, which is a key contribution of this work.

VI. Conclusion

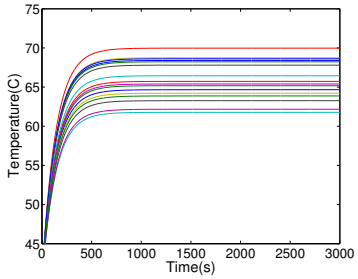
This paper proposes a control-theoretic approach to thermal balancing in server clusters. Our work has the following key contributions: (1) a formulation of thermal balancing as an optimal control problem; (2) a difference equation model that characterizes the thermal dynamics for thermal balancing in server clusters; (3) two thermal balancing algorithms analytically designed based on optimal control theory; and (4) control analysis that establishes the stability and robustness of the control algorithms under system uncertainties in system power consumption. Simulation results demonstrate the capability of our control-theoretic approach to achieve thermal balancing under a wide range of uncertainties in terms of power consumption, ambient temperature, and thermal fault. By employing both temperature and CPU utilization feedbacks in its optimal control design, the CTB-UT algorithm provides a particularly attractive solution for server clusters, as it introduces low control cost for task reallocations and converges quickly to balanced temperatures.

Acknowledgment

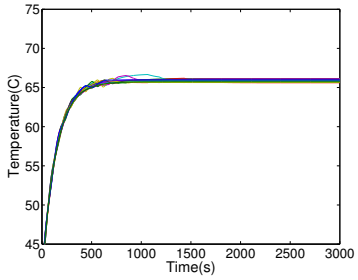
The research of Yong Fu and Chenyang Lu is supported in part by NSF CAREER Award CNS-0448554. The research of Hongan Wang is supported in part by Chinese National Programs for High Technology Research 2009AA01Z337 and Development and Key Project of Chinese National Programs for Fundamental Research and Development 2009CB320804.

References

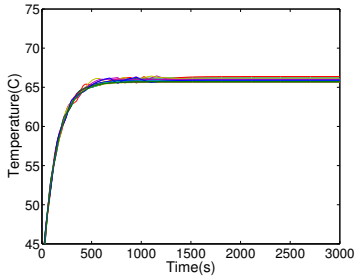
[1] J. Moore, R. Sharma, R. Shih, J. Chase, C. Patel, and P. Ranganathan, "Going beyond CPUs: The portential of temperature-aware solutions for the data center," in *Proceedings of the First Workshop on Temperature-Aware Computer Systems(TACS-1)*, Jun. 2004.



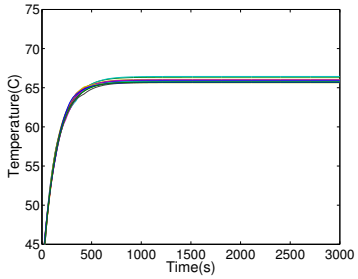
(a) LB



(b) HTB

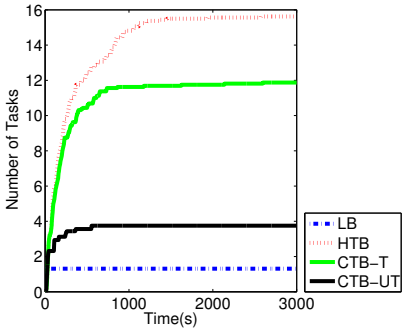


(c) CTB-T

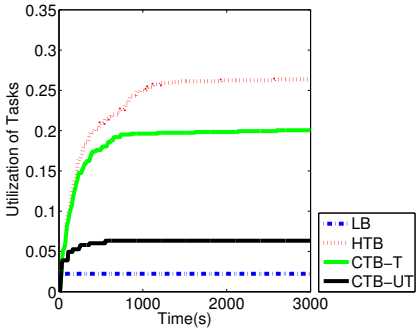


(d) CTB-UT

Fig. 8. Temperatures and CPU utilization of all processors under CTB and baseline algorithms



(a) Reallocated Tasks per Processor



(b) Average Total Utilization of Reallocated Tasks per Processor

Fig. 9. Comparison of Overhead due to Tasks Reallocation

thermal management," in *HPCA*, 2002, pp. 17–28.

[7] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusam, "Compact thermal modeling for temperature-aware design," in *DAC*, 2004.

[8] F. Bellosa, A. Weissel, M. Waitz, and S. Kellner, "Event-driven energy accounting for dynamic thermal management," in *Proceedings of the Workshop on Compilers and Operating Systems for Low Power (COLP'03)*, Sep. 2003.

[9] J. Choi, C.-Y. Cher, H. Franke, H. Hamann, A. Weger, and P. Bose, "Thermal-aware task scheduling at the system software level," in *ISLPED '07: Proceedings of the 2007 international symposium on Low power electronics and design*. New York, NY, USA: ACM, 2007, pp. 213–218.

[10] G. F. Franklin, J. D. Powell, and M. Workman, *Digital Control of Dynamic Systems(Third Editon)*. Menlo Park, CA: Addison Wesley Longman, Inc, 1998.

[11] C. Lu, X. Wang, and C. Gill, "Feedback control real-time scheduling in ORB middleware," in *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'03)*, May 2003.

[12] C. Lu, X. Wang, and X. Koutsoukos, "Feedback utilization control in distributed real-time systems with end-to-end tasks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 6, pp. 550–561, 2005.

[13] R. H. Middleton and G. C. Goodwin, *Digital Control and Estimation: A Unified Approach*. Englewood Cliffs, New Jersey: Prentice Hall, Inc, 1990.

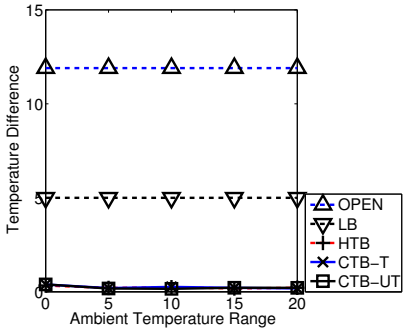
[14] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 1, pp. 46–61, Jan. 1973.

[15] Intel Corp., "Intel Pentium 4 processor in the 423-pin package thermal design guidelines," Intel, Tech. Rep., 2000.

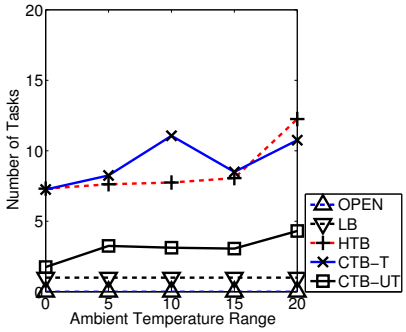
[16] Tom's Hardware, "3.8 GHz P4-570 and e0 stepping to end Intel's performance crisis," Tom's Hardware, Tech. Rep., 2004.

[17] A. P. Ferreira, D. Mosse, and J. C. Oh, "Thermal faults modeling using a RC model with an application to Web farms," in *ECRTS*, 2007.

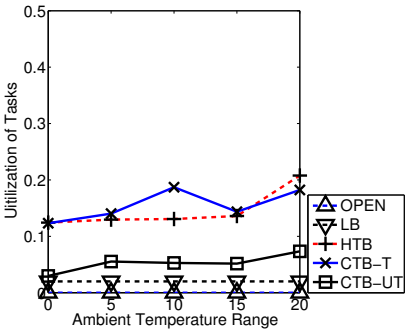
[18] M. Gomaa, M. D. Powell, and T. N. Vijaykumar, "Heat-and-run: Leveraging SMT and CMP to manage power density through the operating



(a) Temperature Difference



(b) Average Number of Reallocated Tasks



(c) Average Utilization of Reallocated Tasks

Fig. 10. Comparison of Algorithms with Different Ambient Temperature. The x axis represents the range of ambient temperatures of processors. For each data point shown in this figure, the ambient temperatures of processors are in the range $[(45 - x/2)°C, (45 + x/2)°C]$.

system," *SIGOPS Operating System Review*, vol. 38, no. 5, pp. 260–270, 2004.

[19] J. Srinivasan and S. V. Adve, "Predictive dynamic thermal management for multimedia applications," in *Proceedings of the Seventeenth Annual International Conference on Supercomputing (ICS'03)*, Jun. 2003.

[20] J. Moore, J. Chase, and P. Ranganathan, "Weatherman: Automated, online, and predictive thermal mapping and management for data centers," in *the Third IEEE International Conference on Autonomic Computing*, Jun. 2006.

[21] A. Merkel and F. Bellosa, "Balancing power consumption in multiprocessor systems," *SIGOPS Operating System Review*, vol. 40, no. 4, pp. 403–414, 2006.