Control-theoretic Thermal Balancing for Clusters

Yong Fu
Department of Computer Science and Engineering, Washington University
fuy@cse.wustl.edu

Chenyang Lu
Department of Computer Science and Engineering, Washington University
lu@cse.wustl.edu

Hongan Wang
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 the *Control-theoretical Thermal Balancing (CTB)* algorithms that employ feedback control loops to dynamically balance the temperature of different servers. CTB features optimal controllers designed based on a difference equation model that captures thermal dynamics of clusters. Stability analysis and simulation results demonstrate that CTB can provide effective temperature balancing when servers' power consumption and ambient temperature deviate from estimation or change significantly at run-time.

Categories and Subject Descriptors

D.4.7 [**Operating Systems**]: Organization and Design—*Distributed systems*

General Terms

Thermal Management, Control-theoretical Thermal Balancing, Distributed Systems, Data Center, Cluster

1. 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* is an attractive thermal management approach for server clusters as it can mitigate hot spots in a cluster by dynamically redistributing load among different processors. In contrast to other thermal management mechanisms such as throttling and dynamic voltage scaling, thermal balancing can avoid overheating a processor without causing performance degradation, which is particularly important for performance-sensitive and real-time applications. Moreover, 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 that distinguishes it from load balancing. First, while a load balancer is designed to balance the *load* on different servers, a thermal balancer aims to balance the *temperature* of different servers. While the temperature of a server is related to its load, the thermal dynamics of a server are more complex, 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 uncertainties and variations in the thermal characteristics, such as varying power consumption, thermal faults, and varying ambient temperature.

To tackle these challenges, we have developed *Control-theoretic Thermal Balancing (CTB)*, a novel thermal balancing approach based on feedback control theory. CTB employs a feedback control loop that periodically monitors the temperature and CPU utilization of different servers in a cluster, and then redistributes clients' service requests among different processors to dynamically balance their temperature.

Specifically, the main contributions of CTB are as follows:

- a difference equation model that characterizes the thermal dynamics of server clusters;
- two CTB algorithms analytically designed based on optimal control theory;
- stability analysis and simulation results that demonstrate CTB algorithms can effectively balance temperature in the face of fluctuations in servers' thermal characteristics.

2. PROBLEM FORMULATION

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

A cluster consists of n homogeneous single-processor servers $\{S_i | 1 \leq i \leq n\}$ connected by high speed 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 on the server.

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 when it is executing tasks. Note the *actual* active power of a processor may deviate from the estimated active power and change dynamically at run time, although only the *estimated* active power is known *a priori*. When a processor is idle, it transits to a sleep state with power, P_{sleep}.

We adopt a widely used thermal model [1] 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, because of the assumption of homogeneous systems, that is:

$$\dot{T'}(t) = AT'(t) + BP(t)$$

where

$$T'(t) = [T'_1(t), T'_2(t), ..., T'_n(t)]^T$$

and

$$P(t) = [P_1(t), P_2(t), ..., P_n(t)]^T$$

while the constant matrices are

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

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

$$\frac{dT'(t)}{dt} = AT'(t) + BP(t) \tag{2}$$

By *bilinear transformation* [3], the continuous form of the thermal model (2) is discretized as

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

where $\Phi = (I + \frac{AW}{2})(I - \frac{AW}{2})^{-1}$ and $\Gamma' = (I - \frac{AW}{2})^{-1}B\sqrt{W}$ and W is the time of sampling interval. Note that in Equation (3) discretized power and temperature, $P_i(k)$, $T'(k)$, which are measured at sampling time kW, substitutes to the continuous power $P_i(t)$ and temperature $T'(k)$.

The thermal balancing problem can be formulated as an optimization problem.

Problem 1 (Thermal Balancing). *The objective of thermal balancing is to minimize the differences among the temperature of different processors, that is,*

$$\min_{k \to \infty} \sum_{i=1}^{n} (T_i(k) - \bar{T}(k))^2 \tag{4}$$

where $\bar{T}(k)$ is the average temperature, defined as $\bar{T}(k) = \frac{\sum_{i=1}^{n} T_i(k)}{n}$.

Based on above objective the design goal of thermal balancing is that the temperature of all processors converge to a same value.

The thermal balancer should also minimize the number of redirections of clients' invocations to different servers. This is important because redirecting clients' invocations can incur non-negligible performance penalty and overhead in many server clusters, e.g., due to loss of cache states or reestablishing the HTTP session.

3. CTB DESIGN AND ANALYSIS

In this section we first provide an overview of *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.

3.1 Overview of CTB

The design goals of CTB are three-fold. First, CTB is designed to balance the temperature of all processors in the clusters. Second, CTB should function robustly under varied power consumption and ambient temperature. Finally, CTB must also reduce the overhead and performance penalty caused by load redistribution.

CTB uses online feedback for thermal balancing. A natural choice of feedback is temperature. However, the temperature responds slowly to load redistribution. As a result, 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 to improve responsiveness of thermal balancing. 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. The feedback control loop dynamically monitors and controls the temperature of individual processor in the system. For the CTB algorithm, the controller input is a vector, $\mathbf{T(k)}$, which includes the temperature of each processor provided by its temperature sensor at the end of 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 invocations that should be redirected among different servers in the following sample period. At the end of the kth sampling period, the feedback loop is invoked and executes the following steps:

1. Each temperature *monitor* sends the controller the controller input based on the measurement over the last sampling period. Most modern processors integrates on-chip temperature sensors. Alternatively, software techniques based on event counters can be used to estimate processor temperature [1]. The utilization monitor measures the CPU utilization in the last sampling period and sends it to the controller. In Linux, the utilization monitor uses `/proc/stat` file to estimate the CPU utilization in each sampling period. The `/proc/stat` file records the number of *jiffies* (each 1/100 of a second) 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 [13].
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 $\Delta U(k)$. In the following sampling period, the balancer will dynamically direct client's service invocations to the servers based on the new tasks allocation.

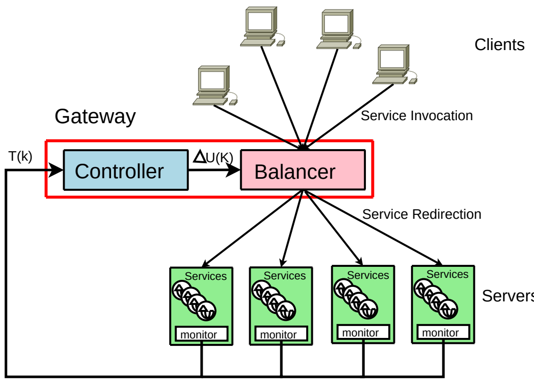


Figure 1: CTB Framework and Components

3.2 Dynamic Model

As the foundation of the control design for thermal balancing, we now derive the dynamic model of a server cluster. We denote the *average* power of the processor Pr_i in the kth sampling period as $\bar{P}_i(k)$. Then we assume for a processor Pr_i, there is a set of tasks, J_i, running on it in the kth sampling period, whose *estimated* utilization $U_i(k)$ is the sum of *estimated* utilization of each tasks, i.e., $U_i(k) = \sum_{j\in J_i(k)} u_j$. The *average* power, $P_i(k)$, can be written as,

$$\begin{aligned}\bar{P}_i(k) &= \tilde{P}_a U_i(k) + P_{sleep}(1 - U_i(k)) \\ &= (\tilde{P}_a - P_{sleep})U_i(k) + P_{sleep}\end{aligned} \tag{5}$$

where $\tilde{P}_a$ is *actual* active power and $G_{u,i}$ is the gain from *estimated* utilization to *actual* utilization. So the average power can be rewritten as

$$\bar{P}_i(k) = G_i(P_a - P_{sleep})U_i(k) + P_{sleep} \tag{6}$$

where the gain $G_i = \frac{\tilde{P}_a - P_{sleep}}{P_a - P_{sleep}}$. By substituting *discrete* power $P_i(k)$ with *average* power $\bar{P}_i(k)$, we can rewrite the thermal model(3) as:

$$T'(k+1) = \Phi T'(k) + \Gamma' G(P_a - P_{sleep})U(k) + \Gamma' P_{sleep} \tag{7}$$

where G is defined as $diag(G_1, G_2, ..., G_n)$. Note the substitution of *discrete* power by *average* power induce approximation error. The analysis shows that approximation error is tolerable in our thermal model. Due to the space limit we omit detailed description of error analysis. Let $T(k) = T'(k) - \tilde{T}$, where $\tilde{T} = \Gamma' P_{sleep}(\Phi - I)^{-1}$, the thermal model of the system can be rewritten as

$$T(k+1) = \Phi T(k) + \Gamma U(k) \tag{8}$$

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

3.3 Control Design of CTB-T

We now detail the design of the controller for the CTB-T algorithm. Recall the control design has two objectives: (1) to minimize the differences among the temperature of different processors, and (2) to minimize the number of reallocated tasks to reduce overhead. The first goal is concerned with thermal balancing, while the second goal is concerned with 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.

$$\min_{k\to\infty} \sum_k [X(k)^T Q X(k)] + \Delta U(k) R \Delta U(k)]. \tag{9}$$

where Q and R are weight matrix respectively and $\Delta X(k)$ is the state of thermal model (8), i.e., $X(k) = T(k)$. If we denote

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

and let $Q = L^T L$, the thermal balancing problem 1 can be transformed to the LQR controller design problem. In fact because difference between processor Pr_i's temperature and the average temperature of all processors is

$$\begin{aligned}\Delta T_i(t) &= T_i(t) - \frac{\sum_{1<k<n} T_k(t)}{n} \\ &= (1 - \frac{1}{n})T_i(t) - \frac{1}{n}T_1(t) \dots - \frac{1}{n}T_n(t),\end{aligned} \tag{10}$$

the first term of optimization objective matches the optimization objective of thermal balancing problem. The second term in the objective is related to the overhead for client redirection. Intuitively, the larger $||\Delta U_i(k)||$, the more clients' invocations must be redirected to/away from the server. We set $R = \rho I$ where ρ is a parameter that can be tuned to achieve the desired balance between thermal balancing and control cost.

3.4 Control Design of CTB-UT

Based on the thermal model (8), there exists significant delay between the change to the utilization and the change to the temperature due to thermal dynamics. To improve the responsiveness of thermal balancer, CTB-UT employs both temperature and utilization as feedback for thermal balancing. CTB-UT also adopts an LQR controller, but the state variables are temperature and utilization rather than only temperature difference.

It is known that dynamics of utilization of processors can be written as [7]

$$U(k+1) = U(k) + \Delta U(k). \tag{11}$$

With thermal model (8), we can present the dynamics of temperature and utilization by

$$\begin{aligned}\begin{bmatrix} T(k+1) \\ U(k+1) \end{bmatrix} &= \begin{bmatrix} \Phi & \Gamma \\ 0 & I \end{bmatrix}\begin{bmatrix} T(k) \\ U(k) \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix}\Delta U(k) \\ &= \Phi_{UT}\begin{bmatrix} T(k) \\ U(k) \end{bmatrix} + \Gamma_{UT}\Delta U(k)\end{aligned} \tag{12}$$

The LQR controller of CTB-UT is derived by temperature and utilization as follows

$$\Delta U(k) = -\begin{bmatrix} K_T & K_U \end{bmatrix}\begin{bmatrix} T(k) \\ U(k) \end{bmatrix} \tag{13}$$

It is noted that in CTB-UT optimization objective of LQR has following form

$$\min_{k\to\infty}\sum_k \left\{ \begin{bmatrix} T(k) \\ U(k) \end{bmatrix}^T Q_{UT} \begin{bmatrix} T(k) \\ U(k) \end{bmatrix} + \Delta U^T(k) R_{UT} \Delta U(k) \right\} \tag{14}$$

where $Q_{UT} = \begin{bmatrix} L & 0 \\ 0 & \lambda \end{bmatrix}^T \begin{bmatrix} L & 0 \\ 0 & \lambda \end{bmatrix}$, $R_{UT} = \rho I$ and λ is a tunable parameter. The first term of optimization objective represents balancing both temperature and utilization. Through tuning λ, we can adjust weight of temperature balancing and utilization balancing. If $\lambda < 1$, the controller is more like CTB-T for thermal balancing. If $\lambda > 1$ the controller is more like load balancing which responds faster to changes to system load. With an appropriate λ, CTB-UT can combine the benefits of thermal balancing and load balancing, by providing effective thermal balancing while responding quickly to system dynamics.

3.5 Stability and Robustness

The stability of a server cluster with CBT is defined as convergence of temperature of all processors to their average temperature. We design the LQR controller 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 remain stable.

THEOREM 1. *The stable range of the G in CTB-T is*

$$diag\left(\frac{1}{1+\sqrt{\alpha}}\right) < G < diag\left(\frac{1}{1-\sqrt{\alpha}}\right), \tag{15}$$

where

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

and W is the time of sampling intervals and S is the solution of Algebraic Riccati Equation (ARE) [3], $S = \Phi^T[S - S\Gamma R^{-1}\Gamma^T S]\Phi + Q$.

The proof of Theorem 1 can be derived directly from Corollary 11.4.1 in [9]. It is noted that since $R = \rho I$ the robust stable region of CTB-T varies along 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 has not a fixed solution like the case of CTB-T. Instead we develop a numerical solution to calculate the robust stability region.

First we derive the closed-loop systems based on thermal model of CTB-UT (12) and optimal controller (13). The closed-loop system has the form

$$\begin{aligned}\begin{bmatrix} T(k+1) \\ U(k+1) \end{bmatrix} &= \left(\Phi_{UT} - \begin{bmatrix} K_T & K_U \end{bmatrix}\Gamma_{UT}\right)\begin{bmatrix} T(k) \\ U(k) \end{bmatrix} \\ &= \Phi_C \begin{bmatrix} T(k) \\ U(k) \end{bmatrix}\end{aligned} \tag{16}$$

For a fixed ρ, Φ_C characterized dynamics of the closed-loop system. According to stability theory, if eigenvalues of Φ_C is in a unit circle the dynamical system is stable. Then varying the value of G and observing eigenvalues of Φ_C we can acquire the stable region of G.

4. EVALUATION

To evaluate the CTB algorithms we developed an event-driven simulator for server clusters. The cluster consists of 16 servers. The temperature of the processor is simulated based on the thermal model (8) with the parameters presented in Table 1. Except thermal capacitance, other parameters are based on Intel technical specification of Pentium IV [6]. The thermal capacitance used here is a result simulated by Hotspot [5].

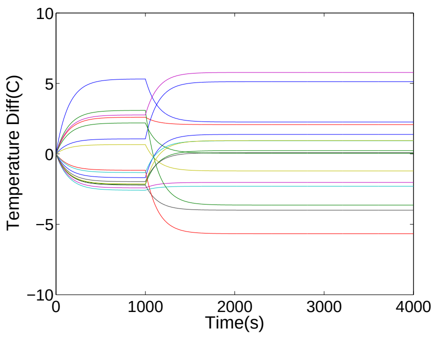
Parameter	Value	Description
Sampling Period(P_s)	4s	
Ambient Temperature(T_a)	45°C	
Active Power (P_a)	51.9 W	
Sleep Power (P_i)	13.3 W	
Thermal Capacity(C)	295.7 J/K	
Thermal Resistance(R)	0.1 K/W	
c_1	0.0034 K/J	$c_1 = \frac{1}{C}$
c_2	0.0338s	$c_2 = \frac{1}{RC}$

Table 1: Simulation Parameters

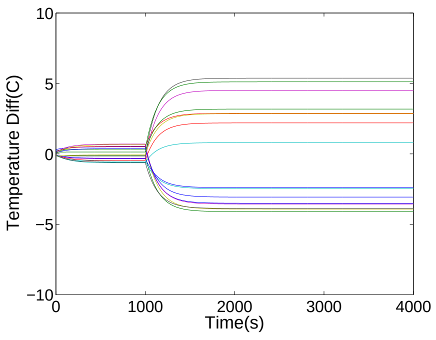
4.1 Baseline Algorithms

We compare the CTB algorithms with an open-loop system (OPEN) without temperature balancing, an Load Balancing (LB) algorithm and a Heuristic Temperature Balancing (HTB) algorithm.

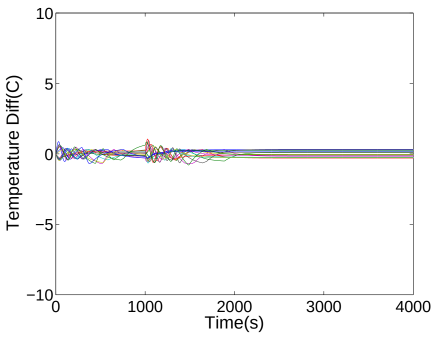
The LB algorithm is designed to dynamically balance the CPU utilization of different processors. The invocation pe-



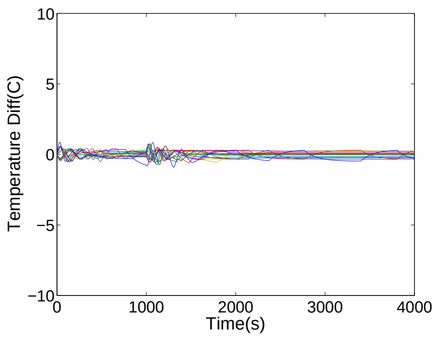
(a) OPEN



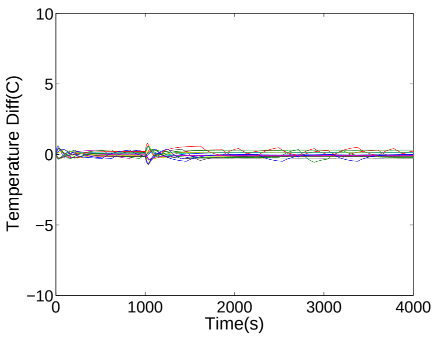
(b) LB



(c) HTB



(d) CTB-T



(e) CTB-UT

Figure 2: Difference in Processor Under Varied Active Power

riod of the load balancing algorithm is the same as the sampling period of CTB. In the end of each period, the algorithm measures the utilization of every processor in the last sampling period and then redistributes the tasks to balance the utilization of different processors in the next sampling period.

Like the CTB algorithms, the HTB algorithm is designed to balance the temperature of different processors based on temperature feedback. In contrast to the control-theoretic approach adopted by CTB, HTB employs to reallocate tasks among the processors. The heuristics is based on the observation that the change to temperature in the steady state is proportional to the variation of utilization of the processor. The imbalance of the temperature between processors can be corrected by migrating tasks whose total utilization is calculated based on the ratio between the changes to the steady-state temperature and utilization. The only difference between HTB and CTB-T is that the controller is computed by LQR in CTB-T while it is a constant matrix present as follows in HTB.

$$K = f_t \begin{bmatrix} 1-\frac{1}{n} & -\frac{1}{n} & \cdots & -\frac{1}{n} \\ -\frac{1}{n} & 1-\frac{1}{n} & \cdots & -\frac{1}{n} \\ \vdots & \ddots & \cdots & -\frac{1}{n} \\ -\frac{1}{n} & \cdots & \cdots & 1-\frac{1}{n} \end{bmatrix},$$

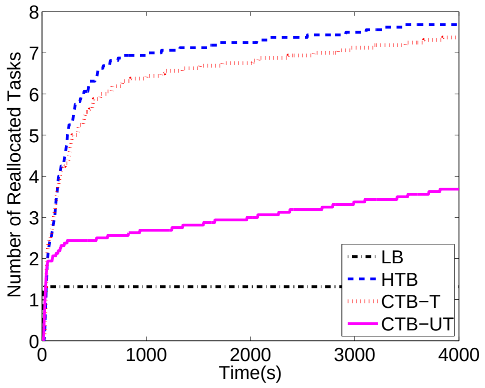
where n is the number of processors and f_t is the factor to represent the amount of utilization necessary to change a unit of temperature. f_t is the ratio between the temperature change and the utilization change in the stead state and can be derived by setting the derivative of temperature as 0 in thermal dynamic equation (1). Note that the HTB algorithm has two key differences from the CTB algorithms: (1) it is not designed to reduce the number of task redistributions (*i.e.*, control cost); and (2) it ignores the thermal dynamics of the system which may influence the transient response to system variations.

4.2 Simulation Results

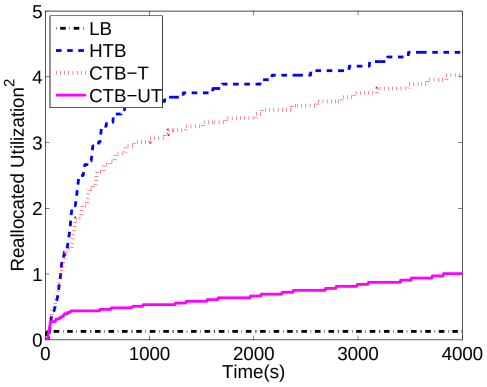
We evaluate the performance of CTB under the cases of varying power. In the first set of experiments we run the simulation for 4000s. At starting time, the utilization of each processor is assigned randomly in the range $[0.4, 0.6]$. We set the parameter of CTB-T as $\rho = 1.0$ and those of CTB-UT as $\rho = 1, \lambda = 0.005$. To evaluate the algorithms' capability to handle changes to the active power, the ratio between *actual* power and *estimated* power, named *power ratio*, of each processor is changed from 1.0 to a random number in range of $[0.8, 1.2]$ at $1000s$. Figure 2 plots the difference between the temperature of each processor and the average temperature of all processors. As shown in Figure 2, after $1000s$ the temperature differences of HTB, CTB-T and CTB-UT converge to close to zero, that is, all three thermal balancing algorithms effectively balance the temperature of the system. In contrast, LB results in significant differences in processor temperature (as high as $11.8°C$) , indicating that load balancing cannot achieve effective thermal balancing among processors with different active power.

We then compare the control cost under the different algorithms. Figures 3(a) and 3(b) show the number and the total utilization of the tasks reallocated by the algorithms.

HTB results in the highest control cost as it does not consider control cost in its design. In comparison, both CTB-T and CTB-UT significantly reduces the control cost, as their LQR controllers are designed to minimize both temperature difference and control cost. In particular, CTB-UT results in the lowest control cost among the thermal balancing algorithms because it responds more quickly to the changes to system states as indicated by its shorter settling time. While LB has even lower control cost, it cannot achieve effective thermal balancing as shown in Figure 2(b). To evaluate the robustness of CTB under varying parameters, the experiment of changing power gain is performed. In Theorem 1 We derive the range of stable gain of CTB-T. In the simulation, G is changed continuously to test the stability of CTB. Figure 4 illustrates the stable region of gain of CTB-T and CTB-UT. Both CTB algorithms maintain stability over wide ranges of power gains. From the figure, we can see CTB is a significantly robust algorithm, 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. Because CTB-UT employs utilization and temperature as feedback to achieve better performance, as a result of trade-off between performance and stability, its stable region is smaller than that of CTB-T. The stable regions of both CTB-T and CTB-UT become wider when ρ increases. Because ρ is a weight of thermal balancing performance relative to control effort in the optimization. When ρ increases, derived LQR controller outputs more significant control output to achieve better performance. Again as a result of trade-off between performance and stability, the stable regions of CTB-T and CTB-UT become wider with ρ increasing.



(a) Number of Reallocated Tasks



(b) Square of Total Utilization Reallocated Tasks

Figure 3: Comparison of Control Cost

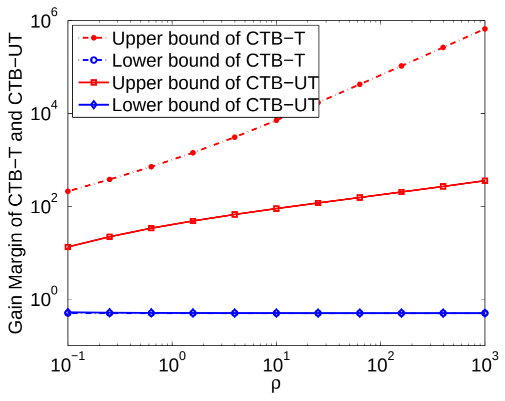


Figure 4: Stable Gain Regions of CTB-T and CTB-UT

5. RELATED WORKS

Feedback-based thermal management has been applied at different levels of computer systems. At the architecture level, the authors of [11] employ feedback control to regulate the instruction fetch rate to control the temperature of the processor. Feedback control is used to manipulate clock gating for mitigating the thermal pressure [2]. Heat-and-Run [4] balances the temperature of different cores in a multicore processor through instruction migration.

At the single-node system level, a proactive method [12] has been proposed to regulate the temperature generated by multimedia applications. At the distributed system level, Weatherman [10] adopts a *data-driven* method to derive the heat distribution in data center and then use this distribution to adjust workload in the data center.

The work most closed related to this paper is Power Balancing [8]. It used an event counter to estimate the power of tasks 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 must be calibrated. Furthermore, their algorithms are not analytically designed based on a control-theoretic approach, which is a key contribution of this work.

6. CONCLUSION

This paper proposes a control-theoretic approach to thermal balancing in server clusters. Our work has the following key contributions: (1) formulation of thermal balancing as a dynamic optimization 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 at low control cost.

Acknowledgment

This research is supported in part by NSF CAREER Award CNS-0448554.

7. REFERENCES

[1] 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)*, Sept. 2003.

[2] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proceedings Of The Seventh International Symposium On High-Performance Computer Architecture*, Jan. 2001.

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

[4] M. Gomaa, M. D. Powell, and T. N. Vijaykumar. Heat-and-run: Leveraging SMT and CMP to manage power density through the operating system. *SIGOPS Operating System Review*, 38(5):260–270, 2004.

[5] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusam. Compact thermal modeling for temperature-aware design. In *DAC '04: Proceedings of the 41st annual conference on Design automation*, pages 878–883, New York, NY, USA, 2004. ACM Press.

[6] Intel Corp. Intel Pentium 4 processor in the 423-pin package thermal design guidelines. Technical report, Intel, 2000.

[7] C. Lu, X. Wang, and X. Koutsoukos. End-to-end utilization control in distributed real-time systems. *IEEE Transactions on Parallel and Distributed Systems*, 16(6):550–561, June 2005.

[8] A. Merkel and F. Bellosa. Balancing power consumption in multiprocessor systems. *SIGOPS Operating System Review*, 40(4):403–414, 2006.

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

[10] 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*, June 2006.

[11] K. Skadron, T. F. Abdelzaher, and M. R. Stan. Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management. In *HPCA*, pages 17–28, 2002.

[12] 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)*, June 2003.

[13] X. Wang, C. Lu, and C. Gill. FCS/nORB: A feedback control real-time scheduling service ofr embedded ORB middleware. *Microprocessors and Microsystems*, 32(8):413–424, Nov. 2008.