Thermal Faults Modeling using a RC model with an Application to Web Farms

Alexandre P. Ferreira and Daniel Mossé
University of Pittsburgh
Computer Science Dept
{apf75,mosse}@cs.pitt.edu

Jae C. Oh
Syracuse University
Electrical Eng & Comp Sci Dept
jcoh@ecs.syr.edu

Abstract

Today's CPUs consume a significant amount of power and generate a high amount of heat, requiring an active cooling system to support reliable operations. In case of cooling system failures, these CPUs can reduce clock speed to prevent damage due to overheating. Unfortunately, when these CPUs are used in a real-time system, a clock control based on frequency-throttling can cause missed deadlines.

In this paper, we first develop and validate a system-wide thermal model that can account for various thermal fault types such as failure of a CPU fan, faults in the case fan and air-conditioning malfunctions. Then we validate the thermal model through experimentation and measurements in AMD Linux boxes.

Our soft real-time power-aware load-distribution algorithm for data centers incorporates a thermal model to minimize the number of missed deadlines that can be caused by thermal faults. We implemented the algorithm in a web-server farm simulator to test the efficacy of thermal-aware load-balancing.

Our results show that the new algorithm helps keep CPU temperatures within the desired thermal envelope, even in the presence of thermal faults. When thermal faults occur, our algorithm improves the QoS, at the expense of higher energy consumption.

1 Introduction

In the mid-90s, many researchers started focusing on power and energy management, to enable long-lived battery-operated devices. Some projects focused on saving energy, and a few of them attempted to control peak power or temperatures in devices. More recently, it has been realized that temperature plays an important role in the performance of applications, in addition to reliability of the microprocessor itself; as popular anecdotal evidence, a video from 2001 on Tom's Hardware site shows what can happen to a CPU that has no *thermal throttling* (Intel vs. AMD) [3]. It has also been shown that temperatures have large financial impact on large companies, because costs with cooling data centers is very high. Temperature-aware scheduling, since then, has been more popular with researchers (in both reliability and energy consumption arenas) and with companies (IBM recently added a data-center reorganization service that deals mainly with power and temperature).

Most of the work done in the area of thermal control has focused on achieving high throughput while maintaining the temperatures below a certain threshold [13]. Clearly, throttling can easily achieve temperature control but with performance impact, and jobs admitted to a real-time system can miss deadlines if the real-time scheduler does not take the throttling mechanism into the account[1]. Accordingly, researchers in the real-time community have started developing schemes that satisfy application deadlines, while maintaining the temperature below the desired threshold [14, 15].

The inspiration for the work reported in this paper comes from two sources. The first is a project run in conjunction with Fermi National Accelerator Laboratories; they provide service to the High Energy Physics (HEP) communities around the world, providing large compute farms to process the vast amounts of data collected. The second is the reported need for power, energy, and thermal controls that go from a single box to a data-center setting, such as Google's installations that contain thousands of machines.

Our first contribution is the development of a mathematical model to study the thermal behaviors of the servers in a data center environment. The model is based on the *resistance-capacitance model* (RC-model) [9] from electrical circuit analysis and design. We have validated our new thermal RC model by comparing it with a Linux box's thermal characteristics. Our second contribution is a *thermal fault model* for servers. The fault model includes failures of cooling fans, can account for external environmental changes and the shutting down of CPUs caused by the throttling mechanism. The third contribution is in showing the application of our thermal model as a building block for

[1] Since frequency has only a linear relationship with power, without DVS, a throttling mechanism will produce a large drop in throughput.

larger projects. Toward that goal, we implemented the two models into a simulator to study the effect of accounting for thermal events on the design of a load-distribution mechanism in a soft-real-time environment, namely a web farm.

Our use of thermal models honor real-time constraints while maintaining the temperature within its specified operational envelope. Our simulation results show that the load-distribution mechanism with thermal consideration also increases performance of the system because it avoids hardware throttling.

This paper is organized as follows: Section 2 presents background research on thermal models. Section 3 presents the thermal model and its validation. Section 4 discusses a thermal fault model. Section 5 describes the model applied to a web farm, Section 6 show the results achieved and finally, Section 7 presents current status and future work.

2 Related Work

There has been a rise in the last five years on thermal management to address the issues of reliability and performance of systems. Cooling requirements are increasing at almost all levels–CPUs, systems, racks and even entire data centers. Most thermal management work has been done in the context of micro-architecture, designing mechanisms to control the temperature inside the chip. These schemes attempt to distribute the workload among the different processing units, so that the chip will be at an even temperature throughout the execution of its workload. A seminal work in this area is the simulation of thermal effects at the architecture level, incorporated into the HotSpot simulator [13].

In [16], a scheduler changes the order jobs are processed in a CPU to achieve a lower temperature, but no DVS schemes are used. In [15, 14] temperature constraints are included in real-time scheduling and dynamic voltage scaling (DVS) is used to achieve a lower CPU temperature, assuming the CPU may achieve very high temperatures during normal operation. In [1], an online approximation algorithm is developed that minimizes the temperature of the CPU.

Data center thermal management also has been studied to mitigate thermal faults, equalize temperature across the data center, and reduce cooling requirements [6, 11]. These efforts consider the server as a black box and consider operation of the rack or entire data center without details of the server itself, which may cause high temperatures at individual servers.

Some researchers have modeled the system with respect to its thermal characteristics. Most use RC models [9] like we do in this paper, while others utilize a more accurate but more complex model based on commercial temperature simulators. Some models focus on the CPU only [12, 16], while others focus on data centers either with cluster implementations or blade servers [7, 4]. Most of these models do not consider thermal faults caused by failures at cooling mechanisms.

Researchers have also been active in thermal management to avoid destruction by excess temperature. CPUs have implemented [8] mechanisms to protect the chip in case of inadequate cooling caused by faults or excess heat production. Some CPU designs allow for overheating but needs active mechanisms to avoid destruction. Most of these mechanisms are not designed to be performance efficient or energy efficient but only provide protection of the CPU. Those are always reactive in the sense that they are activated when the temperature goes above a threshold.

Energy-efficient systems [10] are not necessarily temperature-aware; being more energy-efficient may imply that load is not equally distributed so one server may be hotter. Being temperature-aware allows for the use of a server with a thermal fault (e.g., CPU fan failure) even with reduced capacity, maintaining higher throughput but not overheating it. In [4] a thermal-aware load balancing algorithm is presented but only utilizes turning servers off as the mechanism to manage energy, while we also consider DVS in each server and distribute requests to be more energy efficient.

In summary, we fill a niche for two reasons. First, previous works do not model thermal faults while we consider normal and faulty operation. Second, other works do not consider thermal characteristics (including faults) and energy-efficiency requirements simultaneously, while we combine them for a distributed soft real-time system.

3 The Thermal Model

This section describes the RC-thermal model for real-time server farms. The idea of using RC models for describing temperatures is not new [9], but the novelty of our RC-thermal model is twofold. First, it is able to describe, in a single composable framework, the thermal characteristics of the CPU, the server (containing one or more components), and the entire server farm with a natural composition scheme. Our model uses a small number of elements, which is essential for online computation of the model due to its low complexity, and for using it for prediction of future temperature based on current state and load levels. It is not as accurate as a CFD model [2], but its measured accuracy was within 2^oC for the system tested at all times. Second, our model incorporates not only the heat generators but also the heat absorbers (e.g., the heat sink, the CPU fan, etc) and the behavior of the thermal characteristics of the component when such heat absorbers fail.

We present a validation of the model by comparing it with actual measurements in a Linux box, and show that the mathematical model adequately represents the thermal

characteristics of the Linux farm.

3.1 The RC thermal model

The RC-thermal model is derived from a simple RC-electrical model. Table 1 shows the equivalence of concepts of the thermal RC model and the electrical RC model: we take *current* to represent heat transfer and *current sources* are heat producers. This equivalence is possible because the same equations apply in each model. As an example: Ohms law $R = V/I$ has a equivalent thermal equation as $R_{thermal} = T/H$.

Table 1. *Comparison of Thermal Model to Electrical Model*

Thermal Model	Electrical Model
Temperature T (^oC)	Voltage V(Volt)
Heat Transfer H (Watt)	Current I (Ampere)
Resistance R (^oC/Watt)	Resistance R (Ohms)
Capacitance C (Joule/^oC)	Capacitance C (Coulomb/Volt)
Heat producer	Current source
Constant temperature	Voltage source

A thermal resistance represent the difference in temperature necessary to transfer a certain amount of heat, the unit for resistance is $^oC/W$. Physical properties such as the material types of the components in the system and the size, shape, surface area and the volume of the components have big impacts on this value. A heatsink is designed to have the lowest thermal resistance value so it can transfer a large amount of heat without requiring a large difference in temperature, so it has a large surface area.

In the model, a *capacitor* represents stored heat and it is represented with *Joules* per ^oC, i.e., $J/^oC$. The capacitance measures the amount of energy that is stored or removed to increase or decrease the temperature of an element. Capacitance depends mostly on the heat capacity, type of the material, and the mass of the specific component. For example, a heatsink normally has a large capacitance, whose C (capacitance) value depends foremost on its weight and type of the material (e.g., copper or aluminum).

The above mappings allow us to describe the thermal properties as a system of linear differential equations, a common practice among circuit designers and electrical engineers in general. Our claim is that this very simple thermal RC model, described by a system of differential equations, can predict the dynamic behavior of real-time compute farms, as shown below.

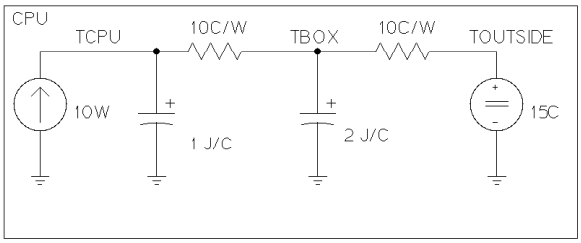

Figure 1. *An example of the thermal RC model.*

Figure 1 shows a simple example with the thermal characteristics of the CPU, box (entire computer), and the outside (room/environment). In the figure, a *node* is the intersection of the lines, depicted as dots near the TCPU, TBOX and TOUTSIDE). In essence, a node is a component or composition of components.

With the following model, we will be able to compute the temperatures at each component, as a function of load and current state, assuming a linear relationship between load and power dissipated. Let t be time, T_i^k temperature at node i at instant k, C_i the thermal capacitance at node i, $R_{i,j}$ the thermal resistance between nodes i and j, $H_{R_{i,j}}^k$ the heat flowing through thermal resistance $R_{i,j}$ at instant k, $H_{C_i}^k$ the heat flowing through thermal capacitance C_i at instant k and H_i as the heat being produced or removed at *node* i.

Equations 1, 2, and 3 are valid for each node in the circuit. Equation 1 represents the conservation of heat, that is, it means that heat cannot be destroyed or created in a node. Equation 2 is the ohms law thermal equivalent. Equation 3 represents the capacitor behavior, that is, the ability of the component to absorb heat.

Using Equation 2, for each node i and all nodes j that a thermal resistance $R_{i,j}$ exists, we can compute $H_{R_{i,j}}^k$. Using Equation 1, $H_{C_i}^k$ can be computed since all values of $H_{R_{i,j}}^k$ have been computed before and H_i is a constant.

$$\sum_j H_{R_{i,j}}^k + H_{C_i}^k + H_i = 0 \tag{1}$$

$$R_{i,j} H_{R_{i,j}}^k = (T_i^k - T_j^k) \tag{2}$$

$$C_i \frac{dT_i^k}{dt} = H_{C_i}^k \tag{3}$$

Equation 4 below is a discretized version of Equation 3. Equation 4 will be used to compute the temperature at instant $k+1$ from the temperature at instant k. Essentially, the model consists of the 3 equations above, and to compute the temperature of a node, this process (equation 4) is repeated iteratively for each interval Δt, for all nodes in the system, until obtaining the temperature at the desired instant or a threshold is reached.

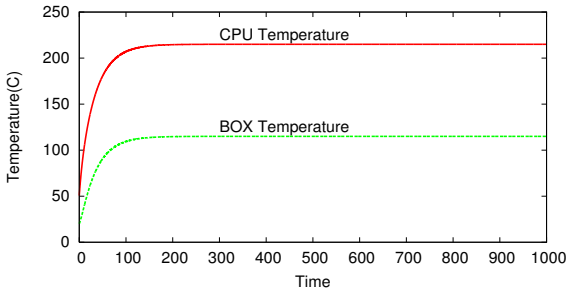

Figure 2. *Temperatures of CPU and case (box) when $C = 2J/^oC$. Compare with Figure 3, and note the rate of temperature increase is much faster.*

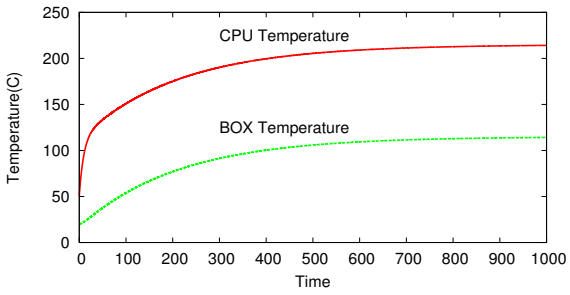

Figure 3. *Temperatures of CPU and case (box) when $C = 20J/^oC$. Compare with Figure 2, and note the rate of temperature increase is much slower.*

$$T_i^{k+1} = T_i^k + \frac{H_{C_i}^k \Delta t}{C_i} \tag{4}$$

To illustrate the effect of the values of the R and C parameters, consider the model in Figure 1, but with two different values of the box thermal capacitance: $2J/^oC$ and $20J/^oC$. In this example, the initial conditions are $T_{cpu} = 50°C$ and $T_{case} = 20°C$. These two scenarios (two box thermal capacitances) represent the rate of temperature increase of the box: the higher the thermal capacitance, the slower the rate of temperature increase, that is, the higher the capacity of the box to absorb heat. In these scenarios, the time constant (i.e., the time it takes for the CPU to achieve its highest stable temperature) changes from 100 seconds to 1,000 seconds as shown in Figures 2 and 3, respectively, even though the final temperature is the same. Note that the difference of temperature between the CPU and box is kept constant after 100 seconds. This shows that for large values of time the CPU capacitance has little effect on the temperature of the CPU, in contrast to the BOX capacitance. This can be used to identify situations such as the CPU temperature as seen in Figure 3: a sharp jump in temperature up to time 10, followed by a slower increase.

3.2 A Compositional Thermal Model

Based on the RC model above for a single component, we compose components of the CPU, the disk, and other parts, to model an entire server. Figure 4 shows a simplified but an accurate model for temperature variations. In this model, the dissipated power (heat generated) of the CPU is not constant but varies between the values of 10.53W to 46W as measured in our server. The power supply has a nominal efficiency of 75% at 300W, so its dissipated power is modeled as a static part (22W), and variable part proportional to the power consumed by the other elements, $0.24 * P_{consumed}$ as shown in Figure 4. To compute the proportionality factor, k, we solve the following equation: PS_dissipated_power = static_part + $k\cdot$ consumed_power, where consumed_power is the power consumed by all components in the computer. In our meaurements, this becomes $300 \cdot .75 = 22 + k \cdot 225$ and thus $k = 0.24$.

The case model represents the air inside the case and its metal frame. Since all other elements CPU, disk, motherboard are inside the case, all their heat has to pass through the case to go to the outside environment. Notice also that in the model for the case, the $12.63W$ source represents the power consumed by other components (e.g., video card and other elements). The power supply is positioned outside the thermal model of the case because it has its own fan that removes heat to the environment directly and not sending hot air through the case, bypassing the case fan and air. The two switches in the figure (labeled Fanl) will discussed in Section 4.

As illustrated above, this model allows us to create any type of systems or data centers. In this case a simple rack that assumes independence of each server is used, so each server has a fixed enviroment temperature of 25^oC.

3.3 Validation

We used a commodity PC to validate the thermal model, with the following specification: Athlon64 3000+ with 512KB cache using AMD cooler, an internal temperature sensor available via *lm_sensors*, 1GB of DDR PC3200 RAM, a motherboard Abit KN8 with Nforce 3 chipset with an internal temperature sensor, a 80GB PATA driver, and a DVD-ROM. The server case has a temperature sensor that can only be read thru an external LCD on the case. Finally, the system runs Gentoo Linux 2006.0, kernel 2.6.16-gentoo-r9.

In order to validate the model, we needed to come up with the values for the constants in the model (the values are shown in Figure 1). For that, we did a first estimation of the constants from the component specifications (when possible and available), from similar components specifications, from the characteristics of the components (e.g., es-

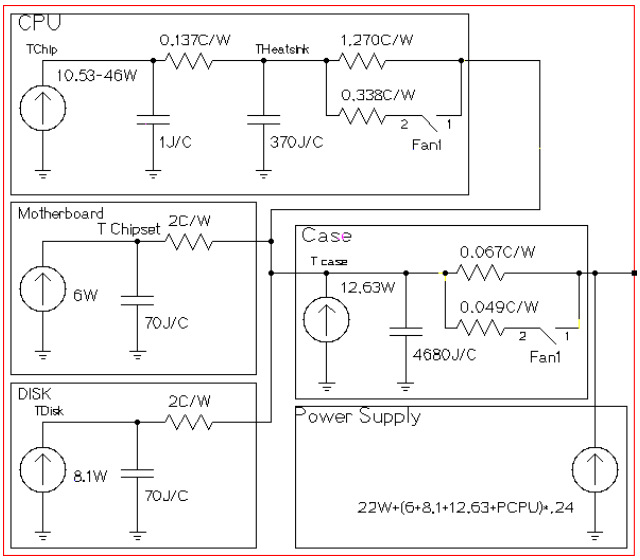

Figure 4. *Thermal model for a server (in a mid-tower case).*

timation of the thermal capacitances of the heatsinks was simple: we weighed the heat sink and computed the value for a 300-gram body of aluminum).

With this first approximation, we created microbenchmarks that increased and decreased the CPU load from 0 to 100% and then from 100% to 0. We then measured (using the instruments described above) the temperatures throughout the experiments. Figures 5 and 6 show the server (CPU and case) temperature when the CPU load goes from 0 to 100% and then when the load is dropped from 100% to 0, respectively.

This process, as any validation process, suffers from the inaccuracies in the model, in the measurements, and in the assigned values of the constants. The mathematical model suggests that the temperature increases and decreases exponentially. We have performed curve-fittings on the collected CPU temperatures and the curve resembles $\Delta T(1 - \exp^{-0.01 \times t}) + T_0$, well matching what the mathematical model suggests.

After the adjustments of the constant values, we experimented extensively with different loads, initial temperatures, environment conditions (colder vs. warmer room temperatures, open vs. closed cases, etc). All the validations matched the mathematical model within 2^oC.

4 Thermal Fault Model

In this section we examine the effects of failures of cooling devices on the temperature of components (directly related components as well as other components). Fault detection is assumed to be done externally to the model, in case of fans by directly monitoring the fan rotation speed.

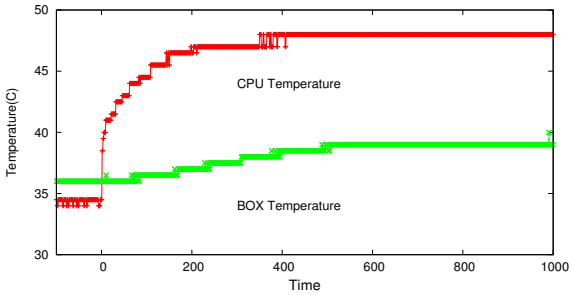

Figure 5. *The system load is increased to 100% from 0% at T=0, and the CPU and the case/box temperatures increase logarithmically.*

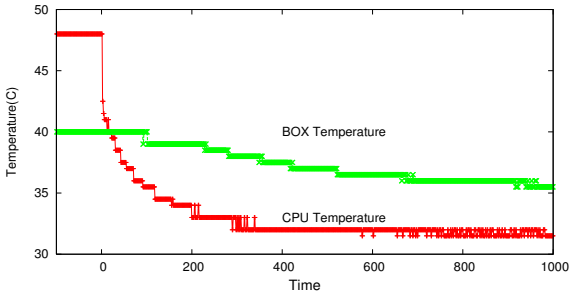

Figure 6. *Temperature variation when the system load is decreased to 0% from 100% at T=0. The CPU and the case/box temperatures decrease logarithmically.*

The thermal RC model suggests different constants when a fault is present and in order to validate the model, we measured the temperatures for the two most common cooling problems: CPU fan failures and case fan failures. These models with faults were compared to the model for normal operating conditions and the difference were mapped. The two cases mapped to only change of value for two specific components of the thermal model: (a) for CPU fan the thermal resistance had its value multiplied by 4.7 when a failure occurs, which means that it is almost 5 times harder to remove heat from the CPU when there is no CPU fan; (b) for the case fan, the value for its thermal resistance doubled, yielding a much less dramatic increase due to the volume of air already in the case. This result is intuitive since the fans are used to increase the amount of heat that can be removed from the heatsinks, a fan failure diminishes this amount. Therefore, we chose to model such faults as a increase of a thermal resistance. To represent it, we added switches in the CPU and case circuits in Figure 4.

The two switches in the figure (labeled Fanl) model the failure of the CPU and case fans, and its correspondent parameters value change. Since the thermal resistance increases when a fan fails, it is represented as a open switch

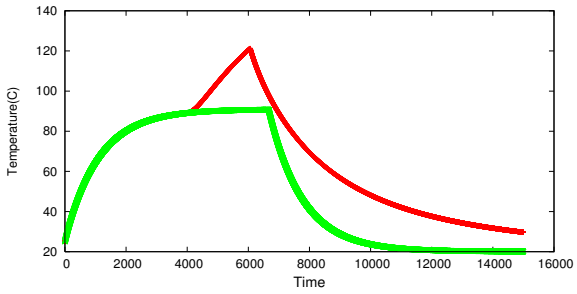

Figure 7. *The temperature behavior when a thermal fault happens and when there is no failures. The top curve depicts a fan failure, increasing the temperature steeply. Even after the load is dropped, the server takes longer to cool down.*

so the value is only one resistor, when the switch is closed two resistors are in parallel decreasing the total value representing the improved heat transfer.

Figure 7 shows an example of the behavior of the temperature in case of a case fan failure. In the graph, at time 4,000, the server has a case fan failure. If no migration is considered (e.g., all other servers have 100% load), the server's temperature increases very quickly as shown. Even when the load is completely removed at around time 6,000 from the faulty server and at time 7,000 for the other servers, it takes a much longer time for the temperature of the fault server to decrease than the others servers since its heat removal capacity is impaired.

Fault detection can also use this model by comparing temperature measurements with values predicted using the model, any discrepancy above certain threshold indicate a problem. Fault isolation is done by not allowing the system to overheat but still using the failed system.

5 Experimental Setup and Models

We performed experiments with the thermal model presented in Section 3.3 and with a real-time power-aware simulator based on [10]. We simulated the operation of a web server farm composed of three homogeneous servers and a front-end. Our objective is to show the dynamic thermal behavior of the servers when cooling failures happen. We kept the size of the cluster small so that the impact of a single server failure can be easily observed with respect to the temperature. Each server is simulated with characteristics described in Table 2.

The load value of 1.0 is the request rate that, on average, consumes 100% of the CPU power. Even though the expected maximum load of the server farm is 3.0 (each server supports individualy a load of 1.0), we defined the maximum load to be 2.5 for our tests. This is because when a

Table 2. *Server simulation parameters*

Parameter	Value
CPU type	Athlon 64 3000+
CPU frequencies	1, 1.8, 2 GHz
AC Standby (Off) server	8W
AC Static power at 1GHz	69W
AC Full load power at 1GHz	76W
AC Static power at 1.8GHz	73W
AC Full load power at 1.8GHz	100W
AC Static power at 2GHz	76W
AC Full load power at 2GHz	113W
Turn-on time	20s
Turn-on energy	1500J
Turn-off time	10s
Turn-off energy	800J
Maximum load without CPU Fan	0.5
Maximum load without Case Fan	0.9
CPU Speed change time overhead	1ms
CPU Speed change energy overhead	20J
Maximum CPU temperature	70°C

failure occurs, a server is only able to support a load of 0.5 (as explained below). Also, following [10], we turn a new machine on when the current set of active servers (those not off or waking up) cannot handle the load. Notice that this is a predictive and conservative mechanism: we turn a new server on before the load saturates the current set of servers, based on the maximum load rate increase allowed for the cluster. In the systems tested, a maximum rate of increase in load is set to 0.3%/sec (load increase of 0.04 every 5sec). At this rate, the amount of load that can appear in the system before another server can booted up is 0.16. This means a new server will be turned on when the load is above 0.84.

The simulator is event based. In order to preserve the accuracy of the thermal model, a maximum time interval is allowed between temperature re-computations (i.e., if no event occurs for a time interval δ, we insert a "compute temperatures" event). The value of δ is calculated to let the temperature to vary at most 1% between events; in our current setup, $\Delta t = 10ms$ (see Equation 4). The thermal-throttling of the CPU was not included in the simulator, to show what happens to the temperature when thermal awareness is not included in the algorithm.

The simulator implements a non-preemptive power-aware load-balancing algorithm [10] with a centralized front-end that distributes requests to the servers. Two power saving methods are used: (a) on-off control at the front-end, to determine the number of servers for the next period based on the load in the previous period; and (b) DVS con-

trol at the local server, emulated by computing the utilization while accounting for awaiting requests and adjusting the speed to keep the utilization smaller than 100%.

The load balancing algorithm uses a power profile for each server. At each period, the load of the last period is used as a estimate for the next period. The load is distributed to the active servers based on the energy efficiency of the servers. Each server has a maximum load it can handle. This method creates a more power-efficient distribution even though it may not divide the load equally.

Application To focus our study on the web-servers, we assume that the front-end load distribution and the back-end database layers are powerful enough, producing no bottlenecks.

In order to verify the thermal fault model, we consider a web-server with a very high request rate–thousands of requests per second. Web page requests are typically composed of a dynamic part and a static part, or a combination thereof, which combine information from different sources. Another similar application is a telephone switch that routes toll-free calls. Toll-free numbers typically get mapped to another landline and therefore a new number must be looked up within a very short deadline in order to route the call correctly. One of the characteristics of our traffic is probably prevalent in most web-sites: a very high variance at CPU consumption for each request and over time. Also, note that the short web-page requests in respect to the thermal time constants ensure that the thermal controls of the type discussed in this paper can be applied at the front-end only, through an admission control and a careful server selection procedure[2].

The traffic applied to the web farm is a mix of both static and dynamic web requests. The execution time of the requests is based on the traffic recorded at `www.cs.pitt.edu`. This traffic was classified into static and dynamic requests. A reference machine, an Athlon 3000+ at 2GHz, was used to characterize the requests in terms of CPU consumption. The static requests were divided into 15 different types with execution times varying from 0.35ms to 0.9ms. Dynamic requests were divided into 5 types from 4ms to 200ms. Each request had its soft deadline defined as the maximum of 50ms or 3 times its execution time. More details can be found in [10].

Power Management Servers can be at standby, idle or active state. In standby state, only a small amount of power is consumed to feed some internal circuits of the server. In the idle state, a server consumes only *static power*, which is the minimum power consumed when the server is able to process requests. A server consumes additional power to process each request. The amount of power consumed is dependent on the type and size of the specific request processed as well as the speed at which the server is operating.

It is well-known that dynamic voltage scaling (DVS) has a large impact on the energy and power consumption of the CPU. Figure 8 shows the impact of DVS on the CPU power evaluated in this paper. Each line in the graph represents a single frequency/voltage operating point, and shows the power consumption as a function of applied load. The power is linear with load, but super-linear with speed (or frequency/voltage). For example, the CPU is approximately twice as efficient at 1GHz with full load, dissipating 17W, than running at 2GHz with 50% max load, dissipating 32W.

Thermal Operation During normal operation, the temperature of the AMD Athlon64 3000+ should not surpass 70°C [5]. One of the characteristics of the processor that can be computed from our model is the maximum load that can be given to the processor when the CPU fan fails.

Recall from Section 4 that the thermal resistance is increased by a factor of 4.7 during a failure. During normal operation at load 1.0, the temperature difference between the CPU and the environment is approximately 23°C (see Figure 5 for maximum temperature and ambient temperature is 25°C, respectively). Since the maximum CPU temperature is 70°C, and the external temperature is 25°C, the maximum difference of temperature is 45°C. Therefore, the difference between normal operation and faulty operation grows by a factor of 1.96.

A reduction of CPU power of a ratio 2.4x (4.7/1.96) is sufficient, and thus, after a CPU fan fails, the maximum power should be no more than 46W/2.4 = 19W (46W is the maximum power dissipated from Figure 9). Load is then determined using Figure 8 and searching for the highest load in a curve below 19W. In this case 1.0GHz at max load of 0.5. DVS is important here because it allows a reduction of power better than linear but with a smaller reduction in performance. Because this processor has a very high static power, DVS becomes more important since a reduction of 2.4 in CPU power would allow a load of only 0.09 at 2GHz or 0.22 at 1.8GHz.

We implemented our thermal model in the simulator and all the servers share the same parameters of the model. The startup and shutdown processes were not simulated: the initial temperature of each server after the system boots is fixed at the measured temperature in the real system as the stable temperature at a idle state.

[2]Long-lived applications are subject of future work, because they may involve migration of existing applications from overloaded/overheated servers.

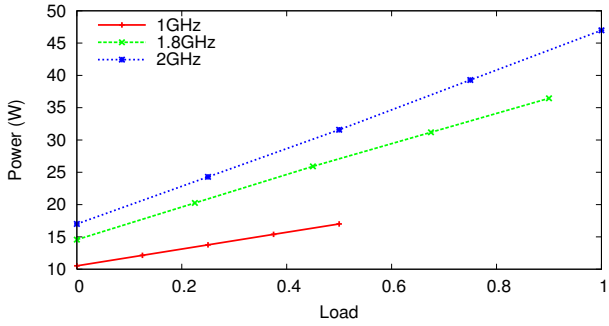

Figure 8. *CPU power measured for different frequencies and supply voltages at maximum system load: at 1GHz, can only process 50% of the load at 2GHz.*

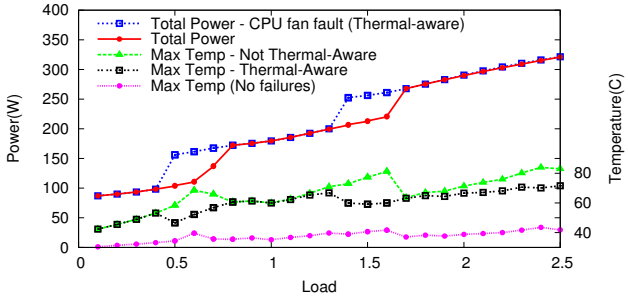

Figure 9. *Total power consumed and maximum temperature of all CPUs versus applied load*

6 Experiments

6.1 Experimental Results

We study the web farm energy and QoS behaviors in the load range from 0.1 to 2.5, with and without the presence of CPU fan failures as thermal faults. QoS is defined in this paper as the percentage of deadlines met among all the requests issued. We assume that all thermal faults occur in server 0, so that the impact of the fault is isolated and observed independently at all nodes. The load-balacing algorithm is unaware of this choice, the only hard constraint is that total load should be less than the sum of maximum load allowed per servers at any point in time. Without this constraint some requests would have been lost, since there is not enough resources available to server them.

Figure 9 presents the total power consumed by all servers and the maximum temperatures of the CPUs given a constant load with and without a single thermal fault present. When a fault is present, the server is assumed to have failed prior to the start of the experiment. Each point is the average of one hour of operation at the fixed load. The power graph has an inclined staircase trend, with two steps. Each step corresponds to a point where a new machine is turned-on. The steps appear because servers have a significant static power, 69W or around 50% of maximum consumed power for a server, and this amount of power is consumed when a machine is turned on and added to the farm. Note that at the average load of 0.7, the increase in "total power" (without faults) consumption is due to the second server being turned on and off intermittently, because of statistical variations that sometimes causes the actual load to be above 0.84.

Figure 9 shows three pairs of temperature (follow right Y axis) and power (left Y axis) curves for each experiment: (1) the operation without thermal failures (curves with circles); (2) with thermal failures but no thermal-aware load balancing (curves with triangles); (3) and with thermal failures and thermal-aware load balancing (curves with squares). The power curves for case (1) and (2) are exactly the same since the thermal-aware portion of the load-balancing mechanism is not active during the experiment.

The temperatures measured in the cluster (see Figure 9) increase with the load until a new machine is turned on, at which point the load per machine is reduced since more servers now are sharing the overall load. The temperature then falls according to the load at each server. The impact of a CPU fan failure is significant: the curves with failure show 20°C to 30°C higher CPU temperature. It is also clear that turning on a new machine, as a result of thermal-aware load-balancing, reduces the maximum temperature by 10°C in the worst case and keeps the temperature consistently below the maximum operating temperature of 70°C.

In Figure 10, we measured the temperatures while keeping a constant load of 0.6 for the duration of the experiments. At $t = 600s$, a CPU fan failure is detected and the front-end is now aware that server 0 can not handle a load of 0.6. Recall that the maximum load allowed with a broken CPU fan is 0.5 as discussed in Section 5. A new server, server 1, becomes operational after its 20-second boot-up time and starts sharing the load. Server 0 now carries a load of 0.4 on average, which is now within its acceptable load, and server 1 is carrying a load of 0.2. The imbalance of load in this case appears as a side effect of the algorithm used in the load distribution.

It is interesting to note that it took almost 1,400 seconds for the CPU at server 0 to reach its final temperature, even though the heat produced at the CPU on average was constant (fixed load). This slow increase in temperature is caused by the large thermal RC (i.e., capacitance and resistance) of the heatsink. This large time constant and the prompt reaction to the fault—taking only 20s to activate a new server—help keep the CPU temperature always

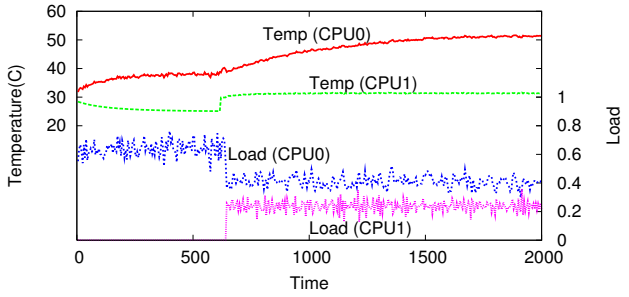

Figure 10. *Effect of CPU fan failure at T=600s over time at fixed load of 0.6*

in a safe zone. Although it has been observed that reducing boot time reduces energy consumption significantly, fast boot times do not significantly affect the maximum temperature of the server. QoS, on the other hand, will be affected because the machine would operate anyway in a lower frequency under DVS.

Each point of the load curve in Figure 10 represents the instantaneous load measured at every 5 seconds. The large variation in load over time is due to the long dynamic web requests, which are responsible on average for 50% of CPU consumption, even though they represent less than 2% of all requests. This high variance can create a situation when a server is overloaded but only for small moments of time even when the average load is well within the sustainable margins on this server. Note that the server is considered overloaded when the peak load hits a 100% CPU utilization at maximum frequency, even if the corresponding average load is smaller. In the simulated case, the maximum load is 1.0 (around 1,000 requests/s) but, since there is slack in the schedule, the server would be able to sustain an instantaneous load of 1.1 (i.e., 1,100 requests/s for the 5 second period measured).

We have performed a more realistic simulation by considering the traces of web requests of a 24h period from jun/30 to jul/01 of the World Cup 98. The traces contain only the number of requests per seconds, type and size transmitted but no information of how much CPU time each request consumes. Dynamic requests are responsible of most of CPU consumption, because static requests are usually cached in servers with large memories and internal bandwidth. We reused the requests distributions from our server, `www.cs.pitt.edu`, but scaled the requests according to the number of requests of the trace. The trace was also scaled so the maximum load was 2.5 (2,500 requests/s).

Figure 11 shows the behavior of the server farm with and without a thermal fault. A single CPU fan fault in server 0 was simulated as occurring at the beginning of the simulation. The temperature graph shows that without any control in the presence of a thermal fault the CPU temperature would be much higher, even surpassing the maximum allowed temperature of 70°C. Using a thermal-aware load-balancing algorithm the temperatur is reduced and is always within the acceptable limits.

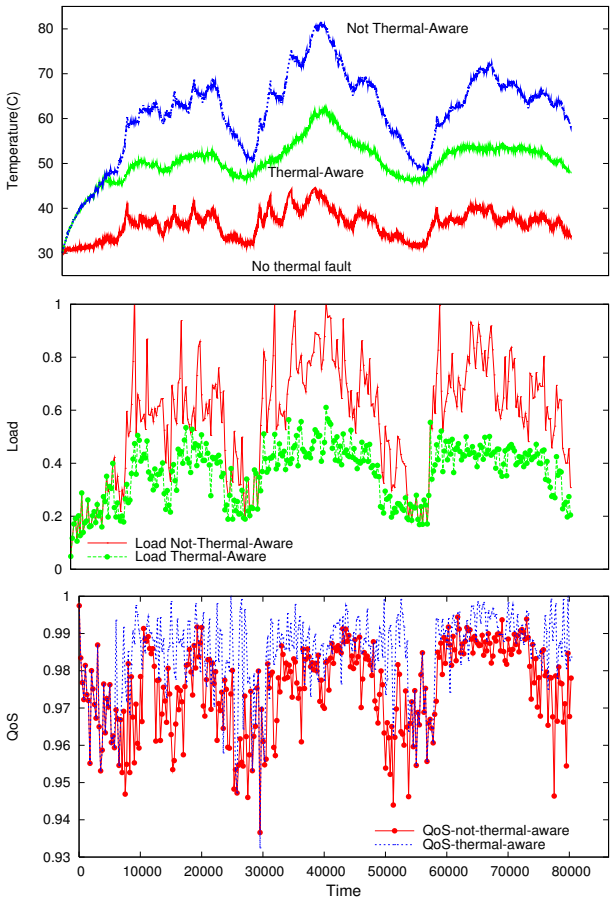

Figure 11. *Temperature, load and QoS (% of met deadlines) for server 0 in simulated web trace*

The load graph of server 0 in Figure 11 shows that the front-end limits the load given to the faulty server to 0.5 (as discussed above) and thus is able to limit the temperature increase at the faulty server. The local DVS in each server is then able to use only more efficient frequencies, since the load is smaller, keeping the temperature at a lower value. Dealing with the fault comes at a price: higher power consumption, since more servers are active to cope with the same load.

The QoS (shown in the bottom graph of Figure 11) is not adversely affected and is even improved, because there are more servers available most of the time to distribute the load, reducing the maximum load to each individual server.

Again, this comes at the cost of higher power consumption. We observed that the QoS of the entire cluster has a similar behavior as the behavior shown for server 0.

7 Current Status and Future Work

In this paper, we presented a mathematical model capable of describing thermal characteristics of server farms. The model can adequately represent thermal characteristics of each component in the system under normal operations as well as under the situations when failures occur in cooling components such as the CPU fan and the case fan. Our model is simple, accurate and easy to compute, opening the possibility of testing thermal impacts of applications on computer systems by fast simulations. This model makes it possible to predict the system temperature at any (future) time instant. Therefore, it allows mechanisms like load-balancing and server switch-on/switch-off to be more dynamic, rather than merely reacting to the situation when the temperature reach a critical point. Furthermore, the algorithm is efficient enough to allow dynamic recomputation of the temperature during runtime.

As a proof-of-concept, we presented a thermal-aware energy-efficient load balancing algorithm that can maintain the operational conditions of the system under cooling system failures. The experimental results show that the algorithm successfully avoided system overheating by distributing the load among the servers achieving fault isolation by mitigating its impact.

In the near future, we plan to improve the model to include variable-speed fans and the effect of temperature over power consumption in newer CPUs. Thermal-aware migration and load-balancing for grid applications are also under investigation. We also plan to study how to do thermal fault detection by comparing expected temperature profile from the model and from actual measurements during runtime. This early fault detection will let us design a mechanism that can avoid unnecessary performance and reliability degradations. Some faults are easily detectable, like fan failures by measuring its rotation speed, but others are not, like filter clogging.

Acknowledgment: This work has been partially funded by NSF/ITR grant; Grant Number: 0325353. We also wish to thank Michael Haney from UIUC for his valueable comments during group meetings at Fermi Labs.

References

[1] J. Chen, C. Hung, and T. Kuo. On the Minimization of the Instantaneous Temperature for Periodic Real-Time Tasks. In *IEEE Real-Time and Embedded Technology and Applications*, 2007.

[2] J. Choi, Y. Kim, A. Sivasubramaniam, J. Srebric, Q. Wang, and J. Lee. Modeling and Managing Thermal Profiles of Rack-mounted Servers with ThermoStat. In *Symposium on High-Performance Computer Architecture (HPCA).*, 2007.

[3] T. Hardware. Hot spot: How modern processors cope with heat emergencies. http://www12.tomshardware.com/images /thg_video_1_cpu_cooling.zip, September 2001.

[4] T. Heath, A. Centeno, P. George, L. Ramos, Y. Jaluria, and R. Bianchini. Mercury and freon: Temperature emulation and management for server systems. In *Conference on Architectural Support for Programming Languages and Operating Systems*, 2006.

[5] http://www.amd.com/. *AMD Athlon®64 Processor Power and Thermal Data Sheet*.

[6] J. Moore, J. Chase, P. Ranganathan, and R. Sharma. Making Scheduling C̈ool: Temperature-Aware Resource Assignment in Data Centers. In *Usenix Annual Technical Conference*, 2005.

[7] J. Moore, R. Sharma, R. S. J. Chase, C. Patel, and P. Ranganathan. Going Beyond CPUs: The Potential of Temperature-Aware Data Center Architectures. In *Workshop on Temperature-Aware Computer Systems*, 2004.

[8] A. Naveh, E. Rotem, M. Moffie, and A. Mendelson. Analysis of Thermal Monitor features of the Intel®Pentium®M Processor. In *Workshop on Temperature-Aware Computer Systems*, 2004.

[9] J. W. Nilsson and S. A. Riedel. *Electric Circuits, Seventh Edition*. Prentice Hall, NJ, 7th edition, 2004.

[10] C. Rusu, A. Ferreira, C. Scordino, A. Watson, R. Melhem, and D. Mosse. Energy-Efficient Real-Time Heterogeneous Server Clusters. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2006.

[11] R. K. Sharma, C. E. Bash, C. D. Patel, R. J. Friedrich, and J. S. Chase. Balance of power: Dynamic thermal management for internet data centers. *IEEE Internet Computing*, 9(1):42–49, 2005.

[12] K. Skadron, T. Abdelzaher, and M. Stan. Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management. In *International Symposium on High-Performance Computer Architecture*, 2002.

[13] K. Skadron, M. Stan, M. Barcella, A. Dwarka, W. Huang, Y. Li, Y. Ma, A. Naidu, D. Parikh, P. Re, G. Rose, K. Sankaranarayanan, R. Suryanarayan, S. Velusamy, H. Zhang, and Y. Zhang. HotSpot: Techniques for Modeling Thermal Effects at the Processor-Architecture Level. In *International Workshop on THERMal INvestigations of ICs and Systems*, 2002.

[14] S. Wang and R. Bettati. Delay Analysis in Temperature-Constrained Hard Real-Time Systems with General Task Arrivals. In *IEEE Real-Time Systems Symposium (RTSS).*, 2006.

[15] S. Wang and R. Bettati. Reactive Speed Control in Temperature-Constrained Real-Time Systems. In *Euromicro Conference on Real-Time Systems (ECRTS)*, 2006.

[16] W. Wu, L. Jin, J. Yang, P. Liu, and S. Tan. Efficient Method for Functional Unit Power Estimation in Modern Microprocessors. In *IEEE/ACM Design Automation Conference*, pages 554–557, San Francisco, CA, Jul 24-28 2006.