2010 39th International Conference on Parallel Processing

System-level, Unified In-band and Out-of-band Dynamic Thermal Control

Dong Li*, Rong Ge**, Kirk Cameron*
*Department of Computer Science, Virginia Tech
**Department of Mathematics, Statistics, and Computer Science, Marquette University
{lid, cameron}@cs.vt.edu, rong.ge@marquette.edu

Abstract

High-density computer racks become increasingly commonplace in supercomputing centers and data centers. With tight integration of high-powered computing components in the racks, hot spots or pockets of elevated temperatures on the chips and system can be easily formed when room air circulation is not effective. Hot spots reduce the reliability of high-density systems and increase the chances of thermal emergencies, which further trigger system slowdowns or shutdowns. Techniques such as dynamically scaling down the voltage of the CPUs and fan control are available on today's systems to reduce heat generation and dissipate heat. Unfortunately, these techniques work independently on their own without cooperation. As a result, to prevent thermal emergencies, systems may work at reduced capacity when full capacity is required. We propose a combined in-band and out-of-band approach to reduce the likelihood of thermal emergency slowdowns and improve the reliability of systems. Our thermal control framework unifies temperature control mechanisms in systems to balance temperature, power consumption, and performance. More precisely, we balance the use of in-band dynamic voltage and frequency scaling (DVFS) with out-of-band proactive fan control. Our results on a power-aware cluster indicate the coordinated use of fan control and DVFS is more effective than either technique in isolation at reducing average system operating temperatures with expected performance.

1. Introduction

Deployments in data centers and supercomputer facilities consist of large numbers of servers that provide computational power for solving challenging commercial, scientific and engineering problems. High-power servers in close proximity to one another generate significant heat that must be removed from the servers and ultimately the facility. Despite the availability of sophisticated cooling systems, hot spots or elevated temperatures in areas of the data center are quite common.

Elevated temperatures cause several problems: First, the additional cooling requirements further stress the HVAC units and increase operational costs. Second, elevated temperatures may cause components (e.g. CMOS circuit, power supply, etc.) to operate less efficiently. Third, high temperatures can trigger thermal emergencies in a server that will slow or shut down the system resulting in severe performance degradation or loss of availability. Fourth, higher temperatures can reduce system reliability and life expectancy.

Previous thermal management techniques, such as dynamic voltage and frequency scaling [2-5] (DVFS) and load migration [6-9], operate *in-band*, i.e., in the critical path of the application execution. In-band techniques can potentially reduce heat substantially. For example, scaling down DVFS processor frequency cubically reduces power consumption and the resulting heat. Nevertheless, the power reduction comes at the expense of the decrease in computation capacity. Less studied are *out-of-band* techniques (e.g. CPU cooling fans [10]) that operate completely outside the critical performance path of an application. Out-of-band techniques cool down hot spots without impacting system computational capacity and application performance. However, relying on cooling fan solely may fail to cool down the hot spots that are already above the emergency threshold.

In-band and out-of-band techniques operate independently without cooperating with each other in today's systems. When sensing thermal emergences, the fans increase the revolution speed (see Figure 1) and the CPUs throttle down. When sensing lower chip temperature, the fans slow down, without recognizing the CPUs are still running at reduced speed. To deliver performance for applications while effectively controlling temperature, we must deploy these two techniques in concert in computing systems.

In-band and out-of-band techniques can coordinate if they work under a unified single controller. The challenges to develop this controller are primarily two-fold. First, in order to coordinate these techniques, we need to be able to enforce the same user control policy across diverse physical mechanisms (e.g., changing CPU frequencies or controlling fan speeds). Second, the controller needs to be easily tunable. It should provide a unified way for the users to explore the tradeoff between control effectiveness and costs for both in-band and out-

0190-3918/10 $26.00 © 2010 IEEE
DOI 10.1109/ICPP.2010.22

of-band techniques. Also it should provide the users with the convenience of choosing either a technique alone, or their combination with superior thermal and performance management.

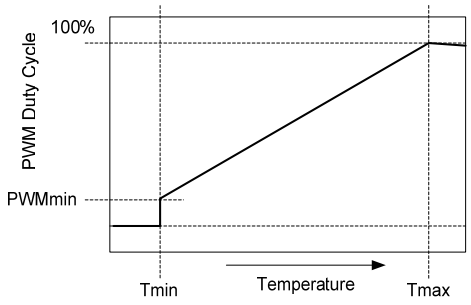

Figure 1: The relationship between temperature and pulse-width modulation (PWM) duty cycle (i.e. fan speed). This picture is adapted from the Analog Devices ADT7467 dBCool remote thermal monitor and fan controller manual [1]. The CPU fan speed is regulated using PWM to control fan revolutions per minute (RPMs).

In this work, we made the following contributions.

- We propose a unified representation that quantifies the effectiveness of various in-band and out-of-band thermal control techniques, which enables coordinated thermal control among them.
- We develop a software framework for system-level, unified in-band and out-of-band dynamic thermal control. We demonstrate that our techniques reduce processor operating temperatures more effectively than DVFS or fan speed control in isolation.
- We explore a number of dynamic thermal management polices and quantify their power, thermal, and performance efficiency.
- We demonstrate that the behavior of parallel applications provides significant opportunities for power and thermal reductions.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 describes our proposed temperature control method. Section 4 presents case studies of our method on CPU fans and DVFS and evaluates the results. Section 5 concludes the paper.

2. Related Work

Thermal management has been studied at various system levels. The work most closely related to ours can be categorized into techniques targeting single components, clusters/servers and data centers. We also discuss formal control techniques for thermal management.

Thermal management at the component level: Many component-level studies such as memory [11-13], disk [14, 15], and microprocessors [16-18] leverage power modes to conserve energy, avoid thermal emergencies, and reduce thermals. Skadron *et al*. [19] proposed component-level, thermal-aware design and power-mode scheduling techniques for processors. In contrast, our work focuses on system-level thermal management and to the best of our knowledge is the first study of the interactive effects of dynamic fan management and DVFS on system thermals.

Thermal management in clusters/servers: Many thermal management techniques address the problem of reacting to or preventing thermal emergencies that lead to system shut downs or malfunctions. Choi *et al.* [10] presented a CFD-based simulation tool for thermal profiling and management. This work considered the use of DVFS in response to fan failure. Ferreira *et al*. [20] simulated a thermal model based on resistance-capacitance. Heath *et al*. [7] designed a thermal emulation tool based on hardware component layout and utilization. Fan failures and other emulated thermal events are used to trigger service request redistributions and admissions. These techniques need careful model verifications which precludes their availability in complex thermal environments. In contrast, our work depends on general thermal control hardware and is portable enough to be used in a large-scale complicated thermal environment. Horvath *et al.* [21] exploited DVFS together with multiple sleep states for the energy management of reconfigurable clusters. Sharma *et al.* investigates adaptive DVFS algorithm in QoS enabled web servers to minimize energy consumption subject to service delay constraints. In essence, this work considers the integration of in-band techniques in isolation; while our work provides a methodology to integrate both in-band and out-of-band techniques.

Thermal management in the Data Center: Moore *et al.* [23] designed Weatherman to predict on-line temperatures using neural networks. They evaluated different workload placements to reduce cooling costs in batch-processing data centers. Mukherjee *et al.* [8] and Samadiani *et al*. [9] used data center thermal distributions to influence system load management and data center design. Ramos *et al*. [24] proposed a software infrastructure for Internet services that selects thermal policy based on dynamic predictions of thermal management technique impact. These techniques primarily consider the isolated effects of workload distribution and dynamic voltage and frequency scaling (DVFS) on system thermals. These approaches do not consider the interactive effects of dynamic fan control on temperature in thermal management of server class machines.

Formal Thermal Control Techniques: Wu and Juang et al. [25] [5] set up a formal model describing a task queue as a proxy for processing loads on cores. They

formally scaled DVFS settings to match varying workload changes. Lefurgy et al. [26] used a closed-loop control to provide precise power control in a real server. Wang et al. [27] used a controller based on multi-input-multi-output control theory to constrain total power of the cluster. These formal techniques usually focus on DVFS in isolation. Our framework can integrate and coordinate different thermal management techniques.

3. Methodology

3.1 Temperature characteristics of parallel applications

Figure 2 shows CPU temperature characteristics observed for parallel applications. We measured a range of parallel workloads [28] and found their thermal behaviors fall into 3 types:

Type I: Sudden change. The CPU temperature increases or decreases drastically over short time periods where the increase or decrease is sustained. This behavior corresponds to drastic or sudden CPU utilization variation.

Type II: Gradual change. The CPU temperature increases or decreases gradually yet steadily over longer time periods at second scale. This behavior corresponds to CPU intensive utilization without proactive temperature control.

Type III: Jitter. The temperature oscillates around a certain value over short time periods. This behavior captures short, bursty CPU utilization. This type differs from the sudden case by the lack of sustained increase or decrease following a spike.

Of these three types, type I and type II lead to actual temperature increase or decrease, while type III does not. Our temperature controller recognizes these types of workload phases in parallel applications and responds accordingly. It proactively controls temperature for type I and II to prevent thermal emergencies, or reduce power consumption when temperature is under control. It is also intelligent not to respond to periods of jitter (Type III).

3.2 History-based context-aware temperature control

Our temperature control methodology is history-based, context sensitive and unified. It (1) periodically profiles temperature and uses the historical information to predict future CPU temperature, and (2) identifies the appropriate technique (DVFS or fan speed regulation) to perform thermal control and balance power and performance for the next interval based on the prediction. Our method is applicable to both out-of-band (e.g. CPU fan) and in-band (e.g. DVFS) temperature regulation techniques.

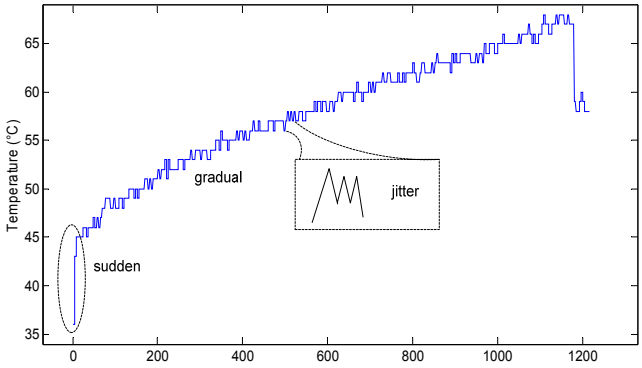

Figure 2: A CPU thermal profile of a system with AMD Athlon64 processor with constant fan speed. Sample rate is 4 samples per second. This profile captures the three most common types of thermal behavior (sudden, gradual, and jitter) observed in parallel applications.

3.2.1 Temperature profiling and prediction

We periodically profile temperature and use a two-level window to track the changes in temperature in both long and short time periods. Level one uses an array to store a small number of most recent temperature samples, and overwrites it for next round of sampling once the array is full. Level two stores the average of the temperatures in level one in a FIFO list, and enqueue and dequeue when a new round of sampling finishes. In such way, the level one window tracks and predicts sudden temperature changes and the level two window tracks and predicts gradual temperature behavior.

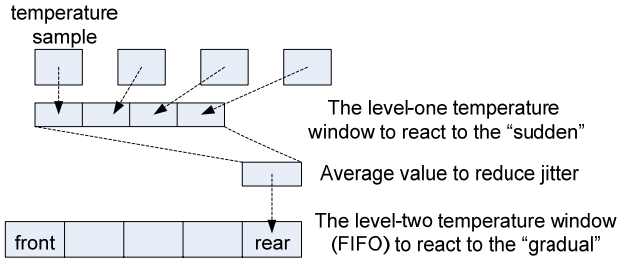

Figure 3: The two-level, history-based, temperature data used in our algorithm to control the effects of sudden, gradual and jitter thermal behaviors.

Figure 3 demonstrates the detailed usage of the two-level window of temperature profiles. We initially fill the level one window with temperature samples. Once the cells are full, we calculate the sums of the samples in the second and first half of the window respectively, and then calculate the difference Δt_{L1} between the sums. A large difference indicates that the processor's work load has recently experienced significant changes. We assume temperature will change with the same rate for the next

round of sampling. The temperature difference is then used to determine the appropriate temperature regulator response. Meanwhile, the cells in this level of window will be cleared out for next round of sampling.

Generally, the array size of level one window is small, yet large enough to capture sudden temperature change and jitter. If the window size is too small, then the controller will react to jitter as if it were a "sudden" sustained behavior. If the window size is too large, then the controller will not promptly respond to sudden sustained behaviors. We experimented with various window sizes and found a 4-entry window was sufficiently large to capture sudden changes and nullify jitter effects for the workloads studied. The window size and sampling rate together determine window update frequency. For example, given a sampling rate of 4 per second and a 4-entry level-one temperature window, the update frequency is 1 second.

To identify trends across larger spans of time, we store the long period and coarse grain temperature profiles in the level two window. Essentially, this window maintains a fixed-size (5 entries in our cases) FIFO queue that stores the average values from the first level. We calculate the temperature difference (Δt_{L2}) between the front and rear items in the queue and use this to predict the temperature behavior over the next interval. More details about how to determine the appropriate temperature regulator response based on the predicted temperature behavior are described in the section 3.2.2.

3.2.2 Target Mode Identification

The identification of appropriate temperature regulation takes two inputs: one is the predicted temperature behavior based on the temperature profile (the section 3.2.1); the other is a parameter specified by the user that indicates the aggressiveness of the temperature controller.

We maintain a *thermal control array* for each available thermal control technique on the system. The control array contains N integer numbers $\{g_1, g_2, \ldots g_n, \ldots g_N\}$, with each number representing a mode that controls temperature at a degree. For example, the thermal control array contains valid CPU frequencies for DVFS control, valid fan speeds for cooling fan, and valid sleep states for ACPI-compatible [32] system. With this common data structure of thermal control array, we are able to unify various in-band and out-of-band thermal management techniques. To enforce a control policy to a single thermal management technique, we vary the values in its corresponding thermal control array. To coordinate multiple thermal management techniques, we fill out the arrays in a unified way.

A value in a thermal control array indicates the effectiveness of this mode to control temperature using the associated in-band or out-of-band technique. For example, a lower CPU frequencies are more effective than a higher frequencies; a higher fan speed is more effectively than a lower one. For a CPU fan with continuous speed, we discretize the speed into a number of distinct speeds; each of the resulting speeds is then a valid mode.

We store the values in the thermal control array in non-descending order by their effectiveness. We also allow duplicated values in the array. An extreme case is that all the values in the array are the same. Herein, the technique associated with the array is not sensitive to temperature changes. N, the bound of the thermal control array, can be equal to or greater than the actual number of available modes on the physical devices such as CPUs and fans.

The actual filled values in the thermal control array is controlled by a user specified parameter P_p. P_p reflects the degree of aggressiveness with which the users want to control temperature. When specifying P_p, the users should be aware of the tradeoffs between temperature management and power consumption or performance delivery. This is because aggressive temperature reduction may cause excessive increased cooling costs: higher CPU fan speeds dissipate heat more quickly while consuming more power; lower DVFS frequencies decrease heat generation more quickly with much more reduced computation capacity. Since P_p reflects a relative degree of proactive control, we use integers within the range of $[P_{MIN}, P_{MAX}]$, i.e., [1, 100] to specify P_p. Controls using larger P_p tend to be cost-oriented, while ones using smaller P_p tend to be temperature-oriented. As we see, P_p is essentially a means of setting control policy.

The details about filling the values into the array is described in the following. The first array element g_1 always stores the least effective temperature control mode, the last element g_N always stores the most effective mode, and the values of the rest array elements are determined by P_p in the following. Given a P_p, we firstly use Eq.(1) to derive an integer number n.

$$n_p = \lfloor (P_p - P_{MIN})(N-1)/(P_{MAX} - P_{MIN}) + 1 \rfloor \quad (1)$$

where P_{MIN} and P_{MAX} is the lower bound and upper bounds of P_P. This integer n_p is a special index of our thermal control array: the array cells with indices in $[n_p, N]$ are filled out with the most effective mode g_N, and the array cells with indices in $[1, n_p\text{-}1]$ are filled out with a subset of physically available modes evenly extracted from the full set. The ratio of n_p to the number of physically available modes on the device determines the subset. If the ratio is less than 1, some physical modes will not appear in the array. If the ratio is 1, then the full set is used to fill out the subarray in $[1, n_p\text{-}1]$. The ratio is

unlikely larger than 1 due to the limitation of available physical modes. A small P_p results in a small n_p. Consequently, 1) more array items store the most efficient temperature mode; 2) a small increment in array index can lead to large increment in cooling effect. Thus the temperature control with small P_p is more aggressive and large P_p is less aggressive.

We use the predicted temperature variations from the two-level window to identify an index in our thermal control array, and the value of the indexed array item is the target mode for the next time interval. Specifically, if the index of the current mode is i, and the temperature variation from the first level of the window is Δt_{L1}, then the target mode index is $(i+c\cdot\Delta t_{L1})$, here c is a constant in the form of $c=(N-1)/(t_{max}-t_{min})$, where t_{min} and t_{max} are lower and upper bounds of safe operating temperature. Such a scheme assures that a temperature changes lead to an appropriate mode change. If the temperature variation from the first level does not result in a change in mode index, our algorithm then uses the temperature variation Δt_{L2} from the second level to see if $(i+c\cdot\Delta t_{L2})$ results in a change.

4. Experimental results and evaluation

Using our thermal control array, our methodology can unify various temperature control mechanisms. We have implemented this methodology on a computation cluster that has user controllable CPU fans with full speed of 4300 RPMs and DVFS capable processors. In this section, we study temperature and power consumption of parallel applications with the unified temperature control on this cluster to examine and evaluate the effects of our methodology.

In the following sections, we evaluate different user control policies with various P_p. Note that we do not mean to pick an optimal P_p for any case in terms of control effectiveness and costs, which is impossible, because an optimal P_p highly depends on application characteristics and system thermal properties. Rather, we mean to develop a tool which has an adjustable parameter P_p to enforce user control policies; we want to evaluate how effectively our system reacts to the P_p in term of power, thermal and performance.

4.1 Experimental platform

The original traditional fan control on the system uses a static mapping between the fan speed and the processor temperature, which is depicted in Figure 1. Fan's RPMs are typically a continuous monotonic function of the PWM duty cycle. In the later discussions, we use the PWM duty cycle to represent corresponding fan speed. The traditional fan speed is set at PWM_{min} when the temperature is no more than T_{min}, and increase linearly with temperature to full speed PWM_{max} when the temperature reaches T_{max}. The parameter values in our cluster are: PWM_{min}=10%, T_{min}=38°C and T_{max}=82°C.

We have implemented a fan driver that dynamically set the fan speed according to both processor temperature and its variation using our methodology. Specifically, we bought an ADT7467 [1] dBCool remote thermal monitor and fan controller from Analog Devices and connected it to the system through PCI interface. We then developed a Linux device driver that regulates fan speed using the i2c [29] protocol. In this driver, we discretize the continuous fan speed into 100 distinct speeds from duty cycle of 1% to 100%, where duty cycle 1% corresponds to the lowest fan speed, and duty cycle 100% corresponds to the highest fan speed. The temperature samples are collected from digital thermal sensors embedded in the processor (AMD Athlon64 4000+) to report core CPU temperature through lm-sensors [30] with a rate of four samples per second. Figure 4 depicts our fan control platform. Embedded remote thermal monitors and fan controllers are common in the motherboard of today's high-end servers. So our method can be easily applied to other servers.

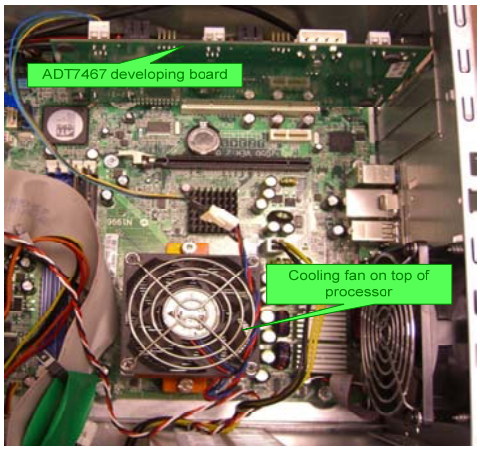

Figure 4: Our fan control Platform.

The processor in our system is DVFS capable and can be scaled among 5 frequencies: 2.4GHz, 2.2GHz, 2.0GHz, 1.8GHz, and 1.0GHz. The processor produces less heat when its frequency is lower. However, our strategy for DVFS control is not to scale down frequency unless necessary because low frequencies impact application performance. Based on this strategy, we implemented a daemon named tDVFS that is directed by our methodology in frequency scaling for temperature control. In this daemon, we trigger frequency scaling when the temperature reaches a threshold.

4.2 Dynamic CPU fan control

We initially run three instances of the cpu-burn [31]

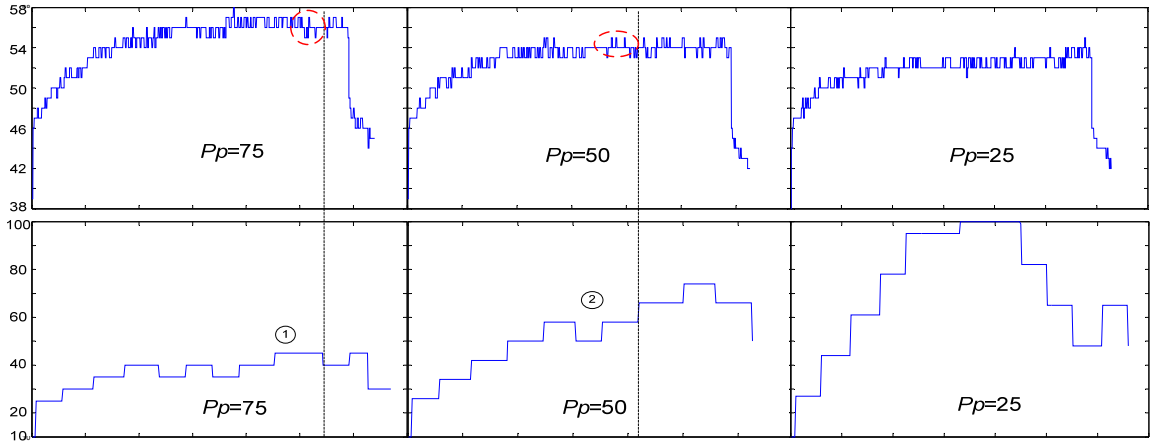

Figure 5: The variations of processor temperature (top) and fan speed (bottom) under our fan control with different P_p (75, 50, 25) The x axis is time and y axis is temperature in °C (top) and PWM duty (bottom). Temperature is sampled every 250ms.

code to test how our dynamic fan control responds to temperature changes. cpu-burn is a program that intensively utilizes the CPU and thus can exhibit a wide range of temperature and patterns. Each run lasts about five minutes. We tested three temperature control policies: 1) aggressive fan control with P_p=25; 2) moderate fan control with P_p=50; and 3) weak fan control with P_p=75. Figure 5 indicates that fan speed with all these three policies quickly responds well to "sudden" temperature variation and as designed does not respond to "jitter" (see ① in Figure 5). Meanwhile, setting P_p to small values leads the lowest temperature, and setting P_p to larger values has less effect and leads to higher operating temperatures. The average PWM duty cycles with the three policies are 36 (P_p=75), 53 (P_p=50), and 70 (P_p=25). Larger P_p values then correspond to lower power use, assuming PWM is directly correlated to power consumption, which it invariably is.

Figure 5 also shows our fan speed control is aware of gradual changes in temperature and responds accordingly. The temperature area marked by a red circle at the left shows a trend to a lower temperature. Our fan speed is thus scaled down to a lower mode, despite the apparent thermal jitter. The temperature area marked by a red circle at the middle shows a trend to a higher temperature. Our fan speed is accordingly scaled up to a higher gear. These proper responses are the results of our history-based thermal analysis.

Next, we compare our dynamic fan control method with the traditional static method and constant fan speed control (the PWM duty cycle is fixed as 75%). The maximum allowed fan speed for traditional fan control and our fan control is also set to 75%. P_p in our fan control is set to 50. Figure 6 shows the fan speed and temperature during the execution of NPB BT.B.4 on four nodes. Fan speed with traditional static control method only reacts to the absolute temperature without foreseeing the increase in the future. As a result, this method needs the longest time to stabilize the temperature yet at the

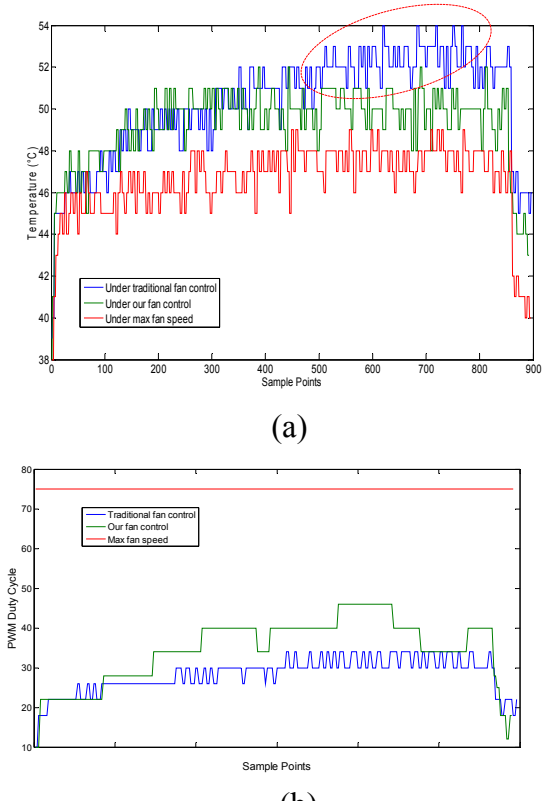

Figure 6: (a) processor temperature and (b) fan speed with our dynamic fan control, traditional static method control, and constant fan speed control. Compared to traditional static method control, ours proactively scales up fan speed and effectively prevents temperature from increasing. Constant fan speed control with high PWM duty cycle maintains the lowest temperature but consumes the most power.

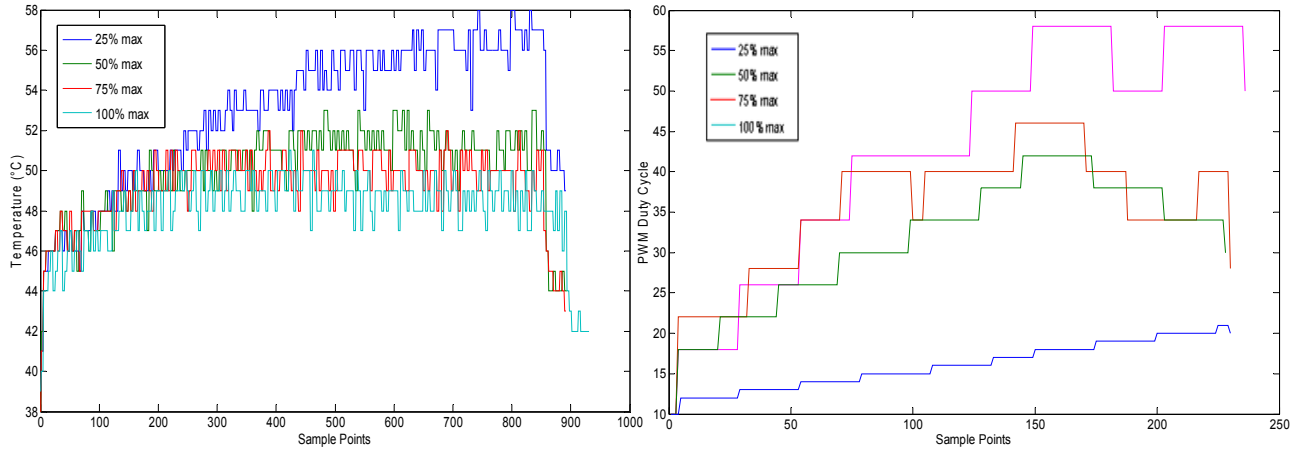

Figure 7: Temperature and fan speed with various maximum PWM duty cycle in our dynamic method. In general, larger maximum PWM duty cycle leads to lower temperature. However, the temperate difference between maximum PWM duty cycles of 50% and 75% is not significant, which implies less powerful fan is able to deliver similar cooling effects as more powerful fan with our dynamic control.

highest degree. In contrast, our dynamic method predicts the temperature increase and thus proactively expedites fan. With our control method, PWM duty cycle increases over 45% against 32% with static method. Consequently, our method stabilizes temperature in a shorter time at a lower degree. Constant fan speed control with high PWM duty cycle maintains the lowest temperature but consumes the most power.

To emulate the cooling effect of different fans, we constrain the maximum PWM duty cycles the fan can revolve in the dynamic control method. Figure 7 depicts the temperature and fan speed with various maximum PWM duty cycles when NAS BT.B.4 benchmark executes on the cluster. P_p is set as 50. The maximum PWM duty cycles are 25%, 50%, 75% and 100% of the full fan speed PWM_{max} respectively. Here a higher PWM duty cycles indicates a more powerful fan. The results show that a more powerful fan brings temperature to lower degrees. The temperature with 100% PWM duty cycle is about 8°C lower than that with 25% PWM duty cycle. However, the results also show there is no significant temperature difference between 50% and 75% maximum PWM duty cycles. This indicates, with proactive fan control methods, a less powerful fan is able to deliver comparable cooling effect as a more powerful one.

4.3 Temperature aware DVFS control

In our experiments tDVFS is coupled with either traditional static or our dynamic fan control. The idea is to use fan to control temperature if possible, and trigger

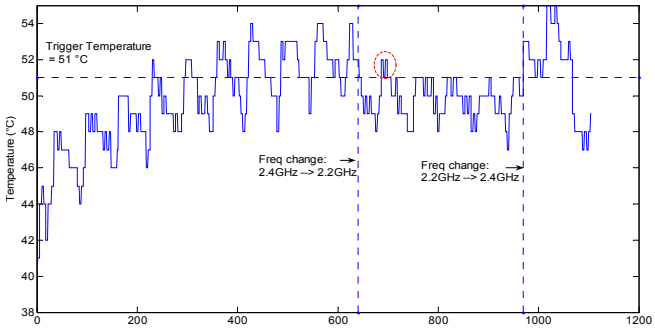

Figure 8: Temperature control with tDVFS. tDVFS scales down processor frequency when average temperature is consistently above threshold and scales processor frequency back up when average temperature is consistently below threshold. The maximal allowed fan speed is 25% PWM duty cycle.

tDVFS to scale down frequency only when temperature is above a threshold. Thus we minimize the impact of frequency down scaling on application performance. The threshold temperature is set as 51°C and P_p is 50 for tDVFS in the following experiments.

We first study the effects of tDVFS when coupled with traditional static fan control. This will help us focus and recognize the dynamics of tDVFS. Figure 8 depicts CPU temperature variation when NAS parallel benchmark LU is executing on four nodes (one MPI

Table 1: Performance and power consumption of BT benchmark when processor speed is controlled by CPUSPEED and tDVFS respectively. tDVFS effectively reduces temperature and power consumption with small performance impact.

	CPUSPEED			tDVFS		
Max allowed PWM duty cycle	75%	50%	25%	75%	50%	25%
# freq changes	101	122	139	2	2	3
Execution Time (s)	219	222	223	219	233	234
Ave Power(Watt)	99.78	99.30	100.80	97.93	94.19	92.78
Power-Delay Product (Watt*s)	21852.78	22044.21	22478.64	21447.27	21946.03	21710.32

process per node). We observe that tDVFS does scale down frequency to reduce temperature when temperature is above a threshold, as expected. However, to minimize the impact on performance, it does so only when average temperature is stabilized above the threshold. Additionally, tDVFS algorithm scales up frequency to its original value once the temperature is consistently below the threshold so as to avoid performance loss. We also highlight an area marked by a red circle in Figure 8 where tDVFS does not respond to short-term thermal behavior and thus avoids performance loss.

Next, we compare the effects of tDVFS with another DVFS daemon named CPUSPEED [33] when both use our dynamic fan control. CPUSPEED periodically adjusts CPU power/performance modes based on CPU utilization during past intervals. Figure 9 shows CPU temperature when NAS parallel benchmark BT.B.4 is executing on 4 nodes. Here P_p in our dynamic fan control is set as 50 and the maximum allowable PWM duty is set to 25%. With these parameters, CPU fan alone is not able to effectively maintain temperature under threshold, and DVFS must act. We observe that the temperature continues to increase when controlled by CPUSPEED, while it is stabilized when controlled by tDVFS. Such results indicate tDVFS is more effective in reducing temperature and preventing hot spots and thermal emergencies.

We have also studied how tDVFS responds to various fan capabilities with the comparison of CPUSPEED, shown in the Table 1. In these experiments, the system power is measured using a Watts up? Pro ES power meter. Compared with CPUSPEED, tDVFS has significantly reduced the number of frequency changes (up to 98.36% as PWM=50%), which is greatly beneficial to the system reliability. When PWM=75%, tDVFS achieve the same performance as CPUSPEED, but uses less power. When PWM=50% and 25%, tDVFS extended the execution time due to the longer execution time at the low frequency than CPUSPEED. However considering both power-saving and performance (i.e., using the metric power-delay product), the tDVFS still performs better than CPUSPEED.

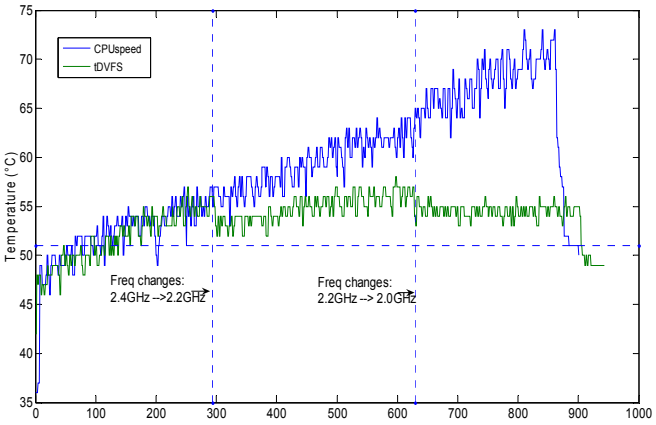

Figure 9: CPU temperature when frequency scaling is controlled by tDVFS and CPUSPEED respectively. Temperature continues to increase with CPUSPEED while is stabilized with tDVFS.

4.4 Dynamic hybrid fan and DVFS control

In the case studies presented above, we have concentrated on either dynamic fan control or temperature aware DVFS. Here we further study how these two mechanisms interact and complement each other in temperature control. In these experiments, we apply an identical P_p value to both dynamic fan control and tDVFS control to reflect user's intention. Figure 10 depicts processor temperature when BT Class B executes on 4 nodes. The maximum allowed PWM duty cycle is set as

50%. The threshold temperature for triggering tDVFS is still set as 51°C.

We first observe that smaller P_p more effectively controls temperature and vice versa, which complies with our design. Second, we notice effective coordination between the out-of-band and in-band thermal control techniques. In particular, the triggering time of tDVFS varies with P_p. The smaller P_p is, the later tDVFS is triggered. This is because dynamic fan control is more aggressive with a smaller P_p, thus effectively maintains temperature at a safe level and avoids the performance loss caused by frequency adjusting. On the contrary, when P_p is larger, if temperature quickly increases above a threshold due to the conservative control of cooling fan, tDVFS is triggered earlier to reduce temperature. Third, we notice the performance differences (i.e., cooling costs) between different P_p. The smaller P_p leads to longer execution time due to its aggressive thermal control policy. In particular, when P_p=25, the CPU frequency is put into the lower frequency than the other cases and has the longest execution time. However we also notice that the performance difference between P_p=25 and P_p=75 is only 4.76%, which means our method tries to minimize the performance impact even though we use aggressive thermal control. In general, our method effectively unifies different thermal control techniques and reacts to different user control policies with minimum performance impact.

5. Conclusion

The thermal control problem is becoming more serious in servers of high-density data centers and supercomputer facilities. However, integrated thermal control for in-band and out-of-band control has not been explored previously. We demonstrate the effectiveness of a thermal control framework that integrates an out-of-band method (fan control) and an in-band method (DVFS). We classified thermal characteristics of parallel applications and used a two-level temperature window to make our controller more effective. We introduced a simple parameter (P_p) to allow the user to specify the aggressiveness of in-band and out-of-band techniques for thermal reductions. This parameter also provides a way to integrate different control methods. We found that using a less powerful fan can achieve the same thermal efficiency as a more powerful fan if we carefully design our fan controller methods. We showed that coordinated use of fan control and DVFS is more effective than either technique in isolation at reducing average system operating temperatures while controlling effects on power and performance.

In future work, we will study how our thermal controllers scale in a large-scale clusters. Also we want to explore the effects of our techniques on OS noise and jitter in scalable systems. In addition, we are considering integration of hardware counter and data in our techniques to improve our prediction mechanisms.

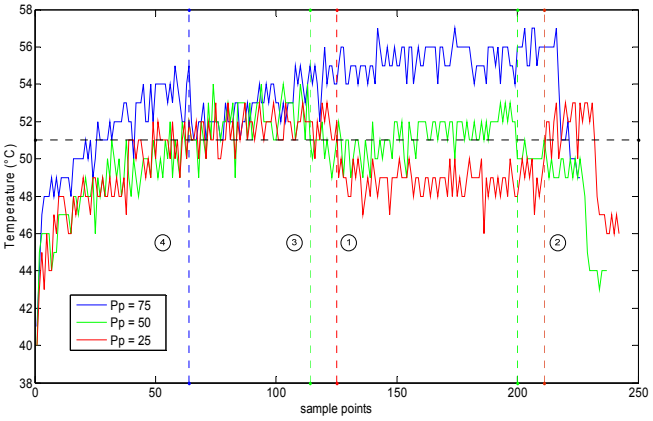

Figure 10: Hybrid temperature control with dynamic fan and tDVFS. Smaller *Pp* leads to lower temperature and triggers frequency scaling later. Frequency scales at ① (*Pp* =25): 2.4GHz→2.0GHz; ② (*Pp* =25): 2.0GHz→2.4GHz; ③(*Pp* =50):

References:

[1] Analog Devices. dBCool Remote Thermal Monitor and Fan Controller ADT7467. http://www.analog.com/UploadedFiles/Data_Sheets/353643468ADT7467_a.pdf

[2] Lin, J., et al. *Thermal Modeling and Management of DRAM Memory Systems*. in *Proceedings of the 34th International Symposium on Computer Architecture (ISCA-34)*. 2007. San Diego, CA.

[3] Donald, J. and M. Martonosi. *Techniques for Multicore Thermal Management: Classification and New Exploration*. in *33rd International Symposium on Computer Architecture (ISCA-33)*. 2006.

[4] Hanson, H., et al. *Thermal Response to DVFS: Analysis with An Intel Pentium M*. in *International Symposium on Low Power Electronics and Design*. 2007. Portland, OR.

[5] Wu, Q., et al., *Formal Control Techniques for Power-Performance Management*. IEEE Micro, 2005. **25**(5): p. 12.

[6] Powell, M., M. Gomaa, and T.N. Vijaykumar. *Heat-and-run: Leveraging SMT and CMP to manage power density through the operating system*. in *Proceedings of the 11th International Conference on architectural support for programming languages and operating systems (ASPLOS)*. 2004.

[7] Heath, T., et al. *Mercury and Freon: Temperature Emulation and Management for Server Systems*. in *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems* 2006.

[8] Mukherjee, T., et al. *Software Architecture for Dynamic Thermal Management in Datacenters*. in *2nd International Conference on Communication Systems Software and Middleware*. 2007.

[9] Samadiani, E. *The thermal design of a next generation data center: a conceptual explosion*. in *1st conference on thermal issues in emerging technologies: theory and application*. 2007.

[10] Choi, J., et al. *Modeling and Managing Thermal Profiles of Rack-mounted Servers with ThermoStat*. in *Proceedings of the 2007 IEEE 13th International Symposium on High Performance Computer Architecture*. 2007.

[11] Ibrahim Hur, C.L. *A Comprehensive Approach to DRAM Power Management*. in *the 14th International Symposium on High-Performance Computer Architecture*. 2008.

[12] Pandey, V., et al. *DMA-aware memory energy management*. in *The 12th International Symposium on High-Performance Computer Architecture*. 2006.

[13] Useche, L., et al. *EXCES: EXternal Caching in Energy Saving Storage Systems*. in *The 14th International Symposium on High-Performance Computer Architecture*. 2008. Salt Lake City, UT.

[14] Kim, Y., et al. *Graceful Operation of Disk Drives under Thermal Emergencies*. in *1st conference on thermal issues in emerging technologies: theory and application*. 2007.

[15] Kim, Y., S. Gurumurthi, and A. Sivasubramaniam, *Understanding the Performance-Temperature Interactions in Disk I/O of Server Workloads*, in *The 12th International Symposium on High-Performance Computer Architecture*. 2006.

[16] Isci, C., G. Contreras, and M. Martonosi. *Live, Runtime Phase Monitoring and Prediction on Real Systems with Application to Dynamic Power Management*. in *the Micro-39*. 2006.

[17] Li, J. and J.e. F.Mart´ınez. *Dynamic Power-Performance Adaptation of Parallel Computation on Chip Multiprocessors*. in *12th International Symposium on High-Performance Computer Architecture*. 2006. Austin, Texas.

[18] Monchiero, M., R. Canal, and A. Gonzalez. *Design Space Exploration for Multicore Architectures: A Power/Performance/Thermal View*. in *In the 20th ACM International Conference on Supercomputing*. 2006.

[19] Skadron, K., et al., *Temperature-aware microarchitecture: Modeling and implementation.* ACM Trans. Archit. Code Optim., 2004. **1**(1): p. 94-125.

[20] Ferreira, A., D. Mosse, and J. Oh. *Thermal Faults Modeling using a RC model with an Application to Web Farms*. in *Proceedings of RTS*. 2007.

[21] Horvath, T. and K. Skadron. *Multi-mode Energy Management for Multi-tier Server Clusters*. in *Proceedings of the ACM/IEEE/IFIP International Conference on Parallel Architectures and Compilation Techniques*. 2008.

[22] Sharma, V., et al. *Power-Aware QoS Management on Web Servers*. in *Proceedings of the 24th International Real-Time Systems Symposium*. 2003.

[23] Moore, J., J.S. Chase, and P. Ranganathan. *Weatherman: Automated, Online, and Predictive Thermal Mapping and Management for Data Centers*. in *Third IEEE International Conference on Autonomic Computing*. 2006.

[24] Ramos, L. and R. Bianchini. *C-Oracle: predictive thermal management for data centers*. in *The 14th International Symposium on High-Performance Computer Architecture*. 2008.

[25] Juang, P., et al., *Formal Coordinated, Distributed Energy Management of Chip Multiprocessors*, in *International Symposium on Low Power Electronics and Design (ISLPED)*. 2005: Bangalore, India.

[26] Lefurgy, C., X. Wang, and M. Ware. *Server-level power control*. in *Proceedings of the 4th IEEE International Conference on Autonomic Computing*. 2007.

[27] Wang, X. and M. Chen. *Cluster-level Feedback Power Control for Performance Optimization*. in *14th IEEE International Symposium on High-Performance Computer Architecture* 2008.

[28] Kirk, W.C., P. Hari K, and S. Varadarajan, *Tempest: a portable tool to identify hot spots in parallel code*, in *Interntional Conference on Parallel Processing*. 2007: Xi'An.

[29] I2C Bus Protocol. http://www.i2c-bus.org/2008. Accessed on Feb 28, 2010

[30] lm-sensors. http://www.lm-sensors.org/. Accessed on Feb 28, 2010.

[31] CPU burn-in. http://users.bigpond.net.au/cpuburn/2008. Accessed on Feb 28, 20010

[32] ACPI: Advanced Configuration and Power Interface. http://www.acpi.info. Accessed on Feb 28, 2010

[33] CPUSPEED. http://www.carlthompson.net/Software/CPUSpeed. Accessed on Feb 28, 2010.

[34] Jin Yang, I. Charles Ume, and Camil Ghiu, and George White, "Board-Level Solder Joint Reliability Study of Land Grid Array Packages for RF Applications Using a Laser Ultrasound Inspection System", 57th Electronic Components and Technology Conference (ECTC), Reno NV, May 2007.