A Framework for Predictive Dynamic Temperature Management of Microprocessor Systems

Omer Khan, Sandip Kundu
Department of Electrical and Computer Engineering
University of Massachusetts Amherst
Amherst, MA 01002
{okhan, kundu}@ecs.umass.edu

Abstract—The sustained push for performance, transistor count and instruction level parallelism has reached a point where IC thermal issues are at the forefront of design constraints. Many of the current systems deploy dynamic voltage and frequency scaling (DVFS) to address thermal emergencies. DVFS has certain limitations in terms of response lag, scalability and being reactive. On the other hand, several hardware based control theoretic schemes have been proposed to deliver optimal performance, but such schemes come at high cost and lack flexibility and scalability. In this paper, we present an alternative thermal monitoring and management system that utilizes software and hardware components, based on virtual machine concept. The proposed scheme delivers targeted, localized, and preemptive thermal management at low cost, adapts well to a multitasking environment, while delivering maximum performance under thermal stress.

I. INTRODUCTION

Power density problems in today's microprocessors are the main cause of hotspots in power-hungry components, such as register files, virtual registers and execution units. Hotspots can lead to circuit malfunction or complete system breakdown. As power density has been increasing with the technology trends, downscaling of supply voltage, and innovations in packaging and cooling techniques to dissipate heat, have lagged significantly due to design and cost constraints.

Several thermal management systems have been proposed as a means to keep a lid on the maximum temperature of a processor [1]-[4]. Among them, Dynamic Voltage and Frequency Scaling (DVFS) is the most well known [5]. Both trigger and management in a typical DVFS system is done through hardware. It has certain drawbacks in terms of response lag and scalability, described with more specifics later in this section.

To address the shortcoming of DVFS scheme, researchers have proposed several control theoretic thermal management schemes that are purely based on hardware to optimally adjust DTM actions for best performance [2]. Such control theoretic approaches work best in a single threaded system and as such suffer from other drawbacks in terms of cost and flexibility.

Ideally, a thermal management solution should push the design to its thermal boundary, while maintaining optimal system performance and throughput. As temperature is well correlated to the application behavior, it is desirable to have insight into process or thread information to guide thermal management in addition to physical triggers like thermal sensors. Avoiding global trigger and response is another key requirement to ensure scalable thermal solution for future many-core designs. In order to tune for best performance at target temperature, thermal solutions need to avoid reactive response to thermal events, with minimal response time. Taken together, these requirements drive our quest for an alternative solution. However, before jumping into any new solution, an evaluation of existing systems with respect to the requirements is in order.

There is an agreed hardware/software framework for power and thermal management through the ACPI framework [6]. When temperature measurements are detected and fed back to the operating system, temporal and/or spatial thermal aware techniques are engaged to eliminate thermal emergencies by reducing power consumption, and performance. While this has been shown to be effective in many situations, ACPI framework is far from perfect. Following are some of the shortcomings of the DVFS/ACPI framework:

Current management techniques are reactive with large response times: On-die droop sensors and thermal sensors are in wide use today. These sensors have inaccuracy problems [7], which, coupled with long system latencies have a detrimental effect on sense-and-react systems. For example, a computer system takes hundreds of micro-seconds to adjust power supply voltage [8]. As a result, sufficient guard band must be put in place to prevent errors from creeping in during the response lag. For future technology trend projections by ITRS, as the power density rises, temperature rises will be faster, voltage and frequency response times that are gated by decoupling capacitor size and PLL lock times will remain similar, and therefore, an even greater guard band has to be used [9].

False Triggers: Usage of greater guard band for sense delays and response lag often triggers false alarms. According to press reports, a large microprocessor design was cancelled in its 5th year due to frequent thermal triggers [10].

Many-core Problems: In a sea-of-core design, power and thermal management is even more problematic. In POWER6 design, there are 24 temperature sensors [11]. If any of these sensors trigger, the response is global. The problem becomes more challenging when there are 100 or 1000 sensors. In that scenario, it is possible that some sensors trigger with alarming frequencies. This will cause a processor to operate mostly at low performance mode.

To slow down only one core, it must have its own clock distribution network. A sea-of-cores with each core having its own PLL and a private clock distribution network is a non-trivial design challenge from a floor-planning and physical design point of view. These are some of the critical challenges for DFS in future. If one core is slowed down, the voltage (DVS) for this core cannot be reduced unless it is in a separate power island. If every core has a separate power island, there will be separate connections between external Voltage Regulator Module (VRM) and the chip, leading to congestion at board level. Consequently, each core will have its own external decoupling capacitor leading to greater power supply noise inside each core. These issues raise the cost of implementing DVS, while the effectiveness of DVS gets reduced. The effectiveness of DFS gets further eroded in 45nm technology, where the power supply voltage is proposed to be 0.9V. For the SRAM bits to work properly, a minimum of 0.7V is needed, reducing the range of supply voltages [9]. Going forward, narrowing of V_{MAX} and V_{MIN} for semiconductor processes will put a lid on DVS.

Other thermal management schemes have been proposed using formal feedback control theory to manage temperature [2]. These schemes are implemented in hardware to adapt to the changing thermal behavior using fine-tuned DTM actions suitable for the hardware to adapt to thermal demands. Although such schemes deliver optimal performance, while maintaining thermal limitations, they are not flexible and lack scalability. The hardware costs of such implementations also make them costly, specifically as we move into the many-core era.

In order to address these issues in thermal management, we propose to unify the thermal monitoring and response management under a common system level virtual framework. Some of the objectives of our proposed framework are:

Scalable thermal management: Insulating the thermal management software from the Operating System, primarily to provide a scalable system level solution. In general, it is not a good idea to involve the OS for thermal management because that requires incremental OS updates as the processor design evolves.

Distributed temperature monitoring: As the chips become larger and feature multiple cores, a targeted response to temperature events becomes necessary. A global response penalizes all threads across all cores. This is not desired, so we move towards distributed sensing and response paradigm.

Action based on rate of change of temperature: A major benefit of the gradient approach is that thermal emergencies can be intercepted before they occur. This allows a smaller temperature guard band. Further, sensors have inaccuracies due to process variation. Multipoint measurements are generally more accurate than single reading, thus alleviating sensor inaccuracy issues.

Low response latency: Coupling predictive actions with low latency response systems can allow a system to operate closer to its temperature limit.

The proposed solution is based on virtualization of the thermal management system and satisfies all of the objectives stated above. Hardware virtualization has become popular in recent times to address a number of problems including booting multiple OS concurrently on a processor, disaster recovery where a copy of a process is in hot stand-by, and server consolidation, to name a few. We decided to extend this approach for thermal management. This forms our core software based thermal management approach. However, there are important benefits of a hardware based approach in terms of real-time performance. So we also propose a second solution: a hybrid scheme combining hardware and software to fine tune actions for thermal management. Finally, we compare simulation based results from these various schemes and conclude that a purely software based approach can be almost as good as hardware assisted software approach. Thus, virtual thermal management is shown to be appropriate and adequate for our goals stated at the onset.

The rest of the paper is organized as follows: In section 2, we discuss the previous work dealing with runtime thermal management. In section 3, we describe our proposed thermal management architecture. Section 4 discusses experimental methodology and section 5 results and analysis. We conclude and provide an insight into our future work in section 6.

II. Previous Work

Our approach tackles the thermal management in a unified hardware/software framework. One of the first hardware, software co-design approach of dynamically managing temperature control was presented in the DEETM framework by Huang et al. [12]. Borkar identified that thermal packaging costs will increase sharply as the chip power budget grows [13]. Dynamic thermal management (DTM) techniques have been proposed to alleviate the thermal packaging costs by enabling the design for temperature less than the peak and use *reactive* DTM to tackle the rare case when temperature limits are approached [1]. The response mechanism initiated by DTM is typically accompanied by degradation in the performance of the chip and persists until normal system operation is resumed. DTM is the philosophy behind many microprocessors thermal design with support for varying levels of operation and fine-grained frequency and voltage scaling [5]. Skadron et al. [2] proposed the use of control theory algorithms for DTM, using fetch gating and migrating computation as their reaction mechanism. Brooks et al. [1] proposed several localized reactive mechanisms – I-cache throttling, decode throttling and speculation control. They also identify dynamic as well as static triggers for these mechanisms. Dynamic triggers may be based on sensors, activity counters or dynamic profiling, whereas, static triggers may be based on compiler optimizations that estimate high-power code segments and inset instructions specifically for DTM trigger.

Rohu and Smith [3] present a software technique that allows the operating system to control CPU activity on a per-application basis. Temperature is regularly sampled and when it reaches a dangerous level, the application (or "hot" process) responsible is slowed down. This technique is shown to be superior to throttling as it does not affect slow processes and it has a better resolution when choosing a slowdown ratio. Srinivasan and Adve [4] proposed a *predictive* DTM algorithm targeted at multimedia applications. They intended to show that predictive combination of architecture adaptation and DVS performs the best across a broad range of applications and thermal limits.

III. Virtual Thermal Management

In this section we describe our unified thermal monitoring and response management scheme under a common system level virtual framework. Our scheme has both hardware and software components. The hardware component consists of thermal sensors that are strategically distributed throughout the chip. Additionally, the microprocessor platform provides reconfiguration capabilities for localized throttling and reduction in capabilities of resources such as queues, buffers and tables, as well as the ability to reduce the width of the major components of the machine such as, fetch, issue and retirement units. We also present some variants of our scheme that incorporate hardware interrupts and some smart control logic for the thermal sensors. Finally, the hardware platform provides support for processor virtualization features like expanded isolation capabilities, and mechanisms for quick thread migration capabilities. The software component of our scheme is the thermal management software that runs natively as a privileged process on the hardware platform. We assume a thin Virtual Machine Monitor (VMM) running underneath the OS software stack, which is primarily used to enter and exit, as well as pass specific thread information to the Virtual Thermal Manager (VTM) [14]. VTM software resides in the physical memory that is concealed from all conventional software including the OS. VTM software maintains thermal history tables for live threads in the system. Based on the thread specific temperature history, and the distributed thermal sensors, the VTM predicts the thermal mapping for the next epoch of computation. When VTM predicts a potential for thermal hotspot(s), it takes an intelligent and preemptive measure by reconfiguring the hardware platform. VTM considers the performance and thermal tradeoffs, and provides adjustments.

Although this approach of managing thermal behavior in software comes at a lower cost and maximum flexibility, it makes conservative assumptions about future DTM actions. Some systems may not be able to tolerate this loss of performance. We present another variant of our scheme, which offloads some of the decision making to hardware. This new variant of VTM is intended to deliver an optimal tradeoff between low cost, flexibility, and the application's execution time. We now distribute the scheduling of thermal actions into a multi-level task. The VTM software takes responsibility of the global scheduling where it determines a range of DTM actions for the hardware platform. These actions are passed on to the hardware platform, which is now assumed to have some support for taking responsibility of managing this reduced set of DTM actions. When the hardware detects that such actions are not delivering within their specified scope, the platform takes an interrupt and invokes VTM software for re-evaluation of DTM actions.

A. *VTM Architecture Framework*

A high level system's view of the VTM architecture is shown in Fig. 1. The operating system passes information about the current and the next thread to the VTM via VMM, based on its scheduling decisions. This is conceivable with the efficient hardware and software support for virtualization. The VMM also maintains a VTM timer that is setup on every VTM exit. This timer is adjusted by the VTM to adjust its sampling to the thermal requirements. The hardware is assumed to have thermal sensors distributed across the platform. These sensors are assumed to register their readings periodically with a thermal controller. The thermal controller can interrupt to invoke VTM and also receives VTM commands.

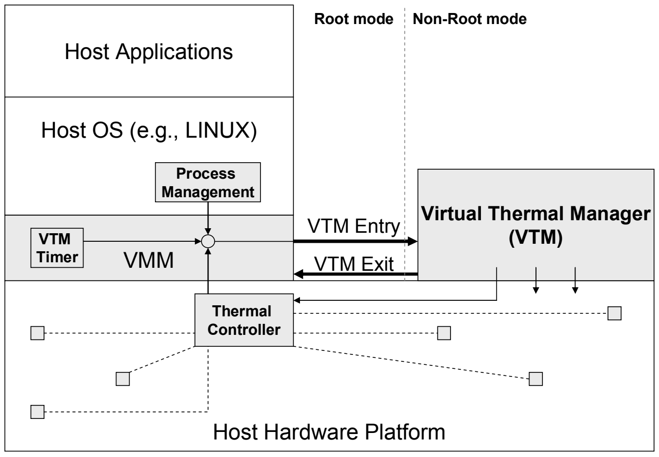

Figure 1. Virtual Thermal Manager's System View

When VTM is active, it has the highest privileged access to the hardware platform. Once VTM software completes its work to determine and setup the DTM actions regarding thermal management, it exits via the VMM and passes control back to the OS. As a result, our approach delivers a hardware-software co-designed solution that assists the hardware to dynamically adjust to tackle the thermal concerns.

B. *VTM Software Details*

The main data structure maintained by the VTM software is the Temperature History Table (THT). THT has an entry for each live thread running in the system. For each thread entry, the THT has an entry for the distributed sensors in the hardware platform. At the lowest level, each THT sensor entry maintains data for the last two temperatures, a 2-bit saturating counter (THC) that tracks the temperature history, running average temperature on a per thread basis, and the last DTM action determined by the VTM. Additionally, VTM tracks the temperature sensor data, which is sampled and updated regularly by the thermal sensors. The high level algorithm for the VTM software is presented in Fig. 2.

```
On each VTM Invocation, and for each Distributed Sensor
  1. Read THT entry for the current & next process
  2. Update last two temperature entries for current process
  3. Determine Projected Temperature for next process, using
       a. Latest temperature reading from the sensor
       b. Last two temperature readings & Average Temp
       c. Temperature history counter (THC) status
  4. Determine DTM Action for next process, using
       a. Projected temperature
       b. Last DTM Action
       c. Last two temperature readings & Average Temp
  5. Setup DTM Action by configuring the hardware platform
  6. Adjust VTM Invocation delay
  7. Update THT entries for THC, Average Temp & DTM Action
```

Figure 2. VTM Software Flow

The 2-bit saturating temperature history counter (THC) is updated based on comparison between the last two temperature readings. The purpose of THC is to capture the long term temperature behavior of each live thread and use it along with the average temperature to predict future temperature.

We use a linear approximation model to project the temperature for the next process. For each sensor, the projected temperature can be calculated by:

$$\text{ProjectedTemp}_{\text{Sensor\#}} = \text{CurrentTemp}_{\text{Sensor\#}} + \alpha$$

The CurrentTemp $_{\text{Sensor\#}}$ is the current temperature status of a particular sensor and α is the projected increase or decrease in temperature for that sensor when the next process will be run by the hardware platform. This projected increase or decrease of future temperature for each sensor is approximated using the last two temperature readings and the THC status. These values are determined based on thermal profiles of workloads.

C. *DTM Action Selection*

Once the projected temperature is calculated, it is used to determine the DTM action for each unit mapped to the sensor. This DTM action is calculated based on the increase or decrease of temperature and the last DTM action for the process to be run next. The selection of DTM actions is dependent on the thermal status of each sensor as well as future application demands. Our analysis of workloads indicates that a targeted, but gradual increase or decrease in DTM action severity levels yields best results in terms of thermal management.

In our proposed system we rank DTM actions in their severity level as shown in Fig. 3. Lower ID indicates a lower severity level. As our proposed scheme's primary focus is to present a thermal management framework, the DTM actions considered here are limited, but enough for the demands of our workloads. Our approach to DTM actions is to initially narrow the processor pipeline to approach a single-issue machine and then use issue throttling to further penalize the processor. Issue throttling of x/y indicates that the processor will operate at full capacity for x cycles and after every x cycles, stall for y-x cycles.

Severity ID	Issue Throttling	Issue Width	Retire Width	Speculation Control
0	1/1	4	4	100%
1	1/1	4	4	75%
2	1/1	2	2	75%
3	1/1	2	2	50%
4	1/1	1	1	50%
5	1/2	1	1	25%
6	1/3	1	1	25%
7	1/4	1	1	25%
8	1/5	1	1	25%

Figure 3. DTM Actions (increasing thermal severity)

D. *VTM Optimiztion with Hardware Assists*

Our software based thermal management scheme may not be able to tackle abrupt thermal changes, as it is limited by the VTM timer's invocation delay. To fix the problem of uncontrolled temperature spikes, we present a simple addition. The thermal controller shown in Fig. 1 invokes an interrupt if the temperature of any sensor exceeds a pre-defined upper threshold level. This interrupt forces the VTM software to re-evaluate the selected DTM action.

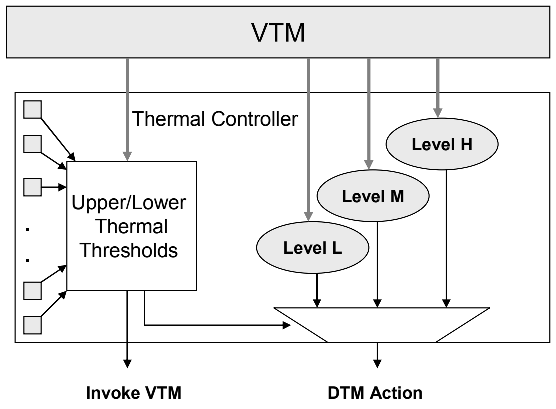

Figure 4. VTM with Hardware Assistance

We realize the benefits of hardware based thermal management and propose a third variant of VTM in this paper. This scheme creates a hybrid implementation of thermal management, where VTM software takes the global coarse-grain responsibilities, but a minimal hardware is added to assist in fine-tuning DTM actions to the changing application demands. Fig. 4 shows the high level structure of the proposed hybrid scheme. VTM determines three levels of DTM actions based on their severity level (Level L is low, Level M is medium and Level H is high) and programs them in special hardware registers. VTM also sets up upper and lower thermal threshold limits for selecting these DTM actions in hardware. The thermal controller periodically samples the thermal sensors and when any of these thresholds are detected, it selects the appropriate DTM action. In case when actions available to the thermal controller are exhausted, it invokes VTM.

IV. EXPERIMENTAL METHODOLOGY

In this section we discuss our simulation environment. We use our modified version of SESC cycle-level MIPS simulator for developing the VTM framework [15]. For modeling dynamic power, SESC implements a version of Wattch [16] and Cacti [17] power estimation tools. We have extended SESC to dynamically invoke HotSpot temperature modeling tool [18]. We have also extended SESC to support DVFS. We have extended SESC to manage and spawn multiple processes dynamically. This allows us to model the VTM entry and exit architecture within SESC.

CPU parameters	
Fetch, Issue, Retire Width	6, 4, 4
L1	64KB 4-way I & D, 2 cycles, LRU
L2	2M 8-way shared, 10 cycles, LRU
ROB Size, LSQ	152, 64
Off-chip memory latency	200 cycles (100 ns @ 2 GHz)

Hotspot parameters	
Ambient Temperature	45°C
Package Thermal Resistance	0.8 K/W
Die Thickness	0.5 mm
Maximum Temperature	85°C
Temperature Sampling Interval	10,000 cycles (of 2 GHz clock)

Figure 5. System Parameters

In context of this paper, we use a single superscalar processor as our hardware platform. System parameters used are shown in Fig. 5. We assume 65nm technology with chip wide Vdd of 1.1V, and frequency of 2.0 GHz. For comparison of VTM, we use DVFS as the DTM response with Vdd of 0.9V and frequency at 800 MHz. Under DVFS, we assume sensor inaccuracy of ±3°C [8], a response latency of 100us, and upper/lower temperature thresholds of 82°C & 80°C respectively. Comparatively, we assume that our approach will result in better sensor accuracy, as multiple temperature readings used to project future temperature, statistically provide more accurate readings. VTM is sampled every 3 to 50ms in our setup, with an estimated entry to exit delay penalty of 150 clock cycles for each invocation. VTM temperature threshold is set to 83°C, assuming a ±2°C error in reading sensors.

For our analysis, we choose SPEC2000 benchmarks. The choice of benchmarks is primarily based on thermal behavior of the program with mcf being *cold*, gcc, gzip, ammp being *hot*, and equake, art, bzip2 being *very hot*. We also run a multi-threaded grouping of these benchmarks. Each benchmark is fast forwarded 2 billion instructions, followed by HotSpot initialization at 76°C for each sensor. This ensures that the processor as well as HotSpot and VTM history tables get sufficient warm up.

For each workload, six simulations are conducted. First simulation, *no-DTM,* is without thermal management. Second simulation is with *DVFS*. Third simulation is based on a control theoretic hardware-only DTM scheme. Fourth, fifth, and sixth simulations are with the VTM framework, with software only categorized as *VTM (SW Only),* software only with hardware interrupt to detect temperature spikes as VTM *(SW + HW UP Interrupt),* and hybrid scheme as *VTM (SW + Smart HW*). Each simulation is run for 1 billion instructions and performance is evaluated based on throughout.

V. Results and Analysis

Fig. 6, Fig. 7, and Fig. 8 show the thermal simulations for a selective number of benchmarks. The data is shown for the worst-case unit i.e., Integer Register File. The x-axis shows the amount of time each run takes for executing the thread and gives an idea of the relative performance for schemes being considered. Fig. 6 shows the hottest workload we have considered for this study. We observe that DVFS in the high activity phase of this application is invoked and is not able to fine-tune to the application behavior, resulting in a constant low performance mode of operation. This leads to an unnecessary loss of performance. On the other hand, all VTM schemes deliver consistently better performance and push the processor closer to its thermal limitations. The software only version of VTM identifies the scenarios when sudden temperature spikes can cause instant and short lived thermal alarms. Our solution, which adds an extra hardware interrupt to invoke VTM when some upper temperature threshold is detected, gracefully handles such thermal spikes. This behavior is more clearly portrayed by the *gcc* workload in Fig. 7.

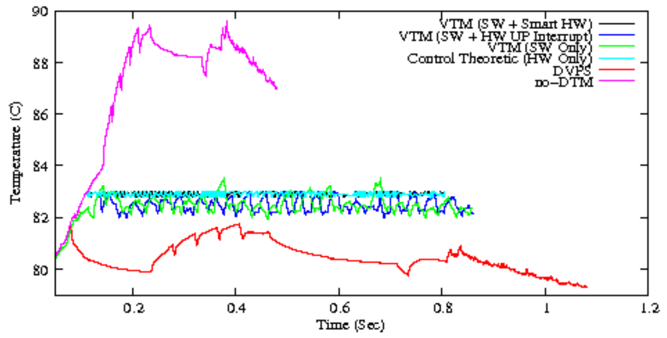

Figure 6. *bzip2* thermal profiles for Integer Register File

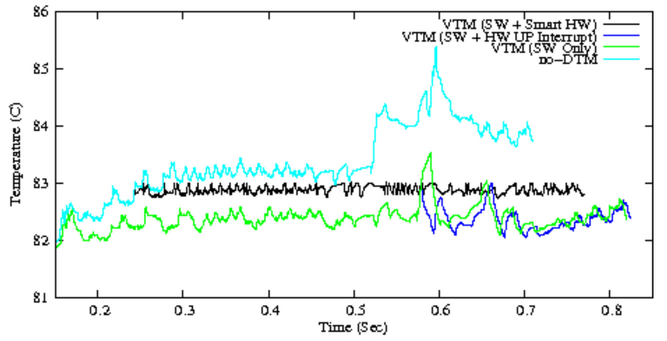

Figure 7. *gcc* thermal profiles for Integer Register File

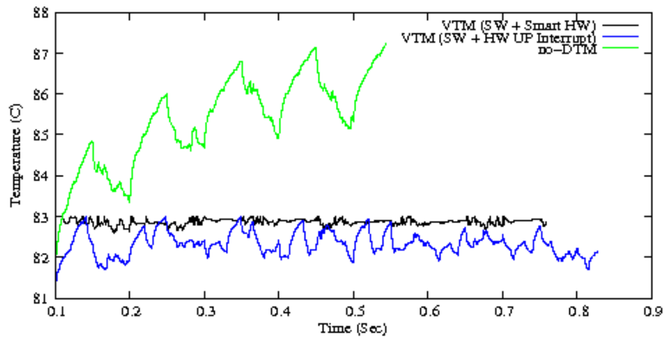

Figure 8. *equake-gcc* thermal profiles for Integer Register File

Fig. 8 shows a multi-programmed workload with two threads running time-multiplexed on a single processor. The thermal behavior of the platform changes when such cooperative scheduling scenarios are considered. Both VTM techniques show the adaptation of VTM to each thread's thermal behavior. VTM with software only, predicts the future trend on a per thread basis and adjusts DTM actions appropriately to maximize performance. The variable VTM invocation policy in our design provides a mechanism for this fine-grain tuning. VTM handles each thread independently, so for *equake* (activity is high) the DTM adjustments are more fine-grain, whereas, for *gcc* (activity is low) the adjustments are tuned to remain in effect for longer periods. This behavior is predominant in Fig. 9, where *art-mcf* workload mixes a hot and a cold thread. The DTM actions are severe and consistently adjusted for *art* and instantaneously attuned for *mcf*.

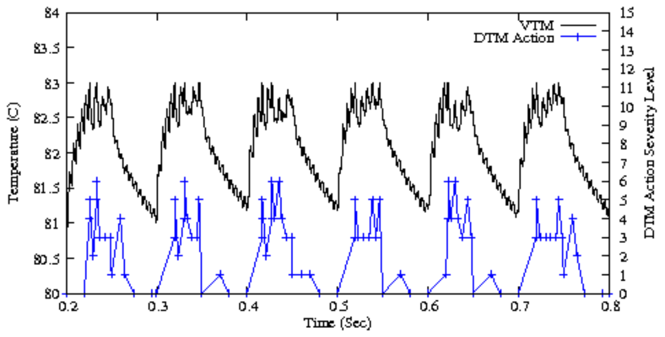

Figure 9. *art-mcf* thermal profile correlated with DTM Actions

Fig. 10 shows the performance impact of our VTM schemes compared to DVFS and a hardware-only based control theoretic scheme. Our data shows that all of the VTM schemes deliver significantly better performance. Specifically, VTM software-only scheme delivers an average of 35%, whereas, VTM with smart hardware delivers an average of 45% improvement over DVFS scheme. The DVFS implementation considered for this study consistently underperforms, mainly due to its limitation of available actions. We consider this fair, as DVFS is predicted to not scale for future technology generations.

On another dimension, we consider a hardware-only based control theoretic scheme, which delivers the best possible performance at targeted temperature. The only limitation imposed on this scheme is the set of available DTM actions based on Fig. 3. This provides a fair comparison with VTM schemes. Fig. 10 shows that VTM with software centric schemes delivers some degraded performance; whereas, the VTM scheme with smart hardware delivers similar levels of performance as the control theoretic scheme. Our VTM scheme with software managed DTM actions is the most flexible, scalable, and cost effective solution for thermal management. The only hardware requirements are the knobs that control DTM actions and temperature sensors. Our data also indicates that VTM by itself acts as a proxy DTM action, when active. This enables our VTM technique to deliver better performance.

Fig. 11 breaks down the execution time into several overheads. The workload overhead is the necessary component consumed by executing the workload. All other overheads are related to thermal management. The DVFS overhead is the 100us overhead of every DVFS invocation, whereas, VTM overhead is the time taken by VTM to compute a DTM action, whenever invoked. The data clearly indicates that VTM and DVFS overheads are a negligible percentage of the total execution time. Specifically, VTM overhead plays minimal role in performance degradation of the processor. This makes the case for using virtual machine as an effective mechanism for thermal management.

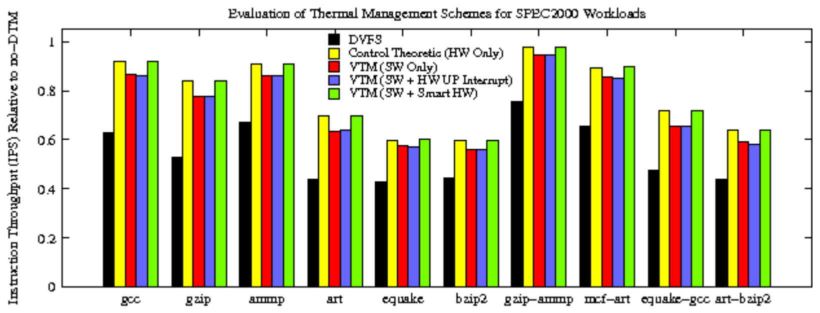

Figure 10. Performance of thermal management schemes for SPEC2000 workloads

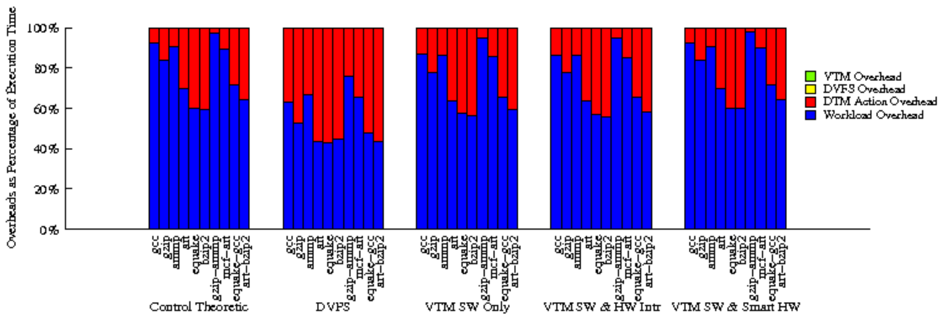

Figure 11. Analysis of overheads of thermal management schemes

We also observe that VTM software-only techniques incur more DTM action overhead compared to the VTM with hardware assist or control theoretic hardware-only schemes. This shows that the limitation of software-only VTM is mainly due to our predictive heuristics being conservative. In the case where hardware interrupt is available to manage temperature spikes, more aggressive DTM actions can be taken based on the thermal history of threads.

VI. Conclusions

We have presented a set of novel thermal management schemes based on using virtual machine to manage DTM action selection and scheduling. The main reason for using virtual machine is its lower overhead cost and flexibility to enable changing thermal demands of many-core designs and applications. We studied VTM as standalone software scheme as well as with hardware assist. The scheme where VTM can delegate some tasks to hardware was found to be the most effective solution. The proposed thermal management scheme adapts to each thread individually to match the computation demands with the hardware thermal limitations. The response time is instantaneous because the proposed scheme is not reliant on PLL lock time and decoupling capacitor charge/discharge time. This scheme delivers a low cost, scalable solution, while keeping the system performance at the edge of thermal boundary.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. 0649824.

References

[1] D. Brooks, M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors", ISCA, 2001

[2] K. Skadron et al., "Temperature-Aware Microarchitecture: Extended Discussion and Results", U. Va., Tech. Report CS-2003-08, 2003

[3] E. Rohou, M. Smith, "Dynamically managing processor temperature and power", In 2nd Workshop on Feedback Directed Optimization, Nov. 1999

[4] J. Srinivasan, S. V. Adve, "Predictive Dynamic Thermal Management for Multimedia Applications", Int'l Conf. on Supercomputing, 2003

[5] T.D. Burd, T. Pering, A. Stratakos, R. Broderson, "Dynamic Voltage Scaled Microprocessor System", ISSCC, Vol. 35, No. 11, Nov. 2000

[6] Advanced Configuration and Power Interface (ACPI). Document available at *http://www.acpi.info/spec.htm*

[7] "Revision Guide for AMD NPT Family 0Fh Processors", AMD Publication # 33610, October 2006, Page 13

[8] C. Poirier, R. McGowen, C. Bostak, S. Naffziger, "Power and Temperature Control on a 90nm Itanium®-Family Processor", ISSCC, 2005

[9] International Technology Roadmap for Semiconductor (ITRS). Document available at *http://public/itrs.net/*

[10] M. Kanellos, "Intel's roadmap quickly reworked", CNET News.com, May 2004

[11] H. Sanchez et al., "Thermal System Management for high performance PowerPC microprocessors", Proc. of COMPCON, Page. 325, 1997

[12] M. Huang, J. Renau, S. Yoo, J. Torellas, "A Framework for Dynamic Energy Efficiency and Temperature Management", MICRO-33, 2000

[13] S. Borkar, "Design Challenges of Technology Scaling", In IEEE Micro, Vol. 19, Issue 4, Pages 23-29, 1999

[14] Jim Smith, Ravi Nair, "Virtual Machines: Versatile Platforms for Systems and Processes", Morgan Kaufmann Pub, 2005

[15] J. Renau et al., "SESC Simulator", 2005; *http://sesc.sourceforge.net*

[16] D. Brooks, V. Tiwari, M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations", ISCA, 2000

[17] P. Shivakumar, N. Jouppi, "CACTI 3.0: An integrated cache timing, power, and area model", Tech. Report 2001/2, Compaq, 2001

[18] K. Skadron et al., "HotSpot: Techniques for Modeling Thermal Effects at the Processor-Architecture Level", In THERMINIC, 2002