Hardware/Software Co-design Architecture for Thermal Management of Chip Multiprocessors

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

***Abstract*—The sustained push for performance, transistor count, and instruction level parallelism has reached a point where chip level power density issues are at the forefront of design constraints. Many high performance computing platforms are integrating several homogeneous or heterogeneous processing cores on the same die to fit small form factors. Due to the design limitations of using expensive cooling solutions, such complex chip multiprocessors require an architectural solution to mitigate thermal problems. Many of the current systems deploy Dynamic Voltage and Frequency Scaling (DVFS) to address thermal emergencies, either within the Operating System or hardware. These techniques have certain limitations in terms of response lag, scalability, cost and being reactive. In this paper, we present an alternative thermal management system to address these limitations, based on hardware/software co-design architecture. The results show that in the 65nm technology, a predictive, targeted, and localized response to thermal events improves a quad-core performance by an average of 50% over conventional chip-level DVFS.**

I. INTRODUCTION

Power density problems in today's microprocessors have become a first-order constraint at run-time. 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. These problems are further exacerbated for small and restricted form factors.

Ideally, a thermal management solution is expected to push the design to its thermal boundary, while delivering optimal system performance. As temperature is well correlated to the application behavior, it is desirable to have an insight into thread information to guide thermal management in addition to physical triggers like thermal sensors. Avoiding global trigger and response is another key requirement to ensuring scalable thermal solution for future many-core era. In order to tune for best performance at target temperature, thermal solutions need to deliver predictive response to thermal events, while keeping the response time and cost overheads low.

There is an agreed hardware software framework for power and thermal management through the ACPI framework [1]. 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 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, which, coupled with long system latencies have a detrimental effect on sense-and-react systems. For example, a computer system takes 100s of micro-seconds to adjust clock frequency and reduce power supply voltage [2]. Additionally, a large manufacturer had a product recall for server parts in 2006 due to sensor inaccuracy [3]. As a result, sufficient guard bands 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, a greater guard band has to be used [4].

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 [5]. 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 frequency. 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 the 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 the 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 DVS 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 [4].

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

978-3-9810801-5-5/DATE09 © 2009 EDAA

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 OS change as the processor design changes.

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.

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 safety margin, also known as temperature guard band. Further, sensors have inaccuracies due to process variation. Multipoint measurements are generally more accurate than single reading. This can potentially alleviate 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.

Application Adaptation: A tight correlation exists between temperature and application behavior. Adapting to each thread's thermal demands can optimize system performance, while keeping the cores thermally saturated.

The proposed solution is based on virtualizing 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.

The rest of the paper is organized as follows: In section 2, we discuss related work and provide motivation for our proposed architecture. In section 3, we describe our proposed thermal management architecture. Section 4 discusses experimental methodology and section 5 results and analysis. We conclude in section 6.

II. Related Work

Our approach tackles thermal management in a unified hardware/software framework. One of the first hardware software co-design approaches of dynamically managing temperature control was presented in the DEETM framework by Huang et al. [6]. Borkar identified that thermal packaging costs will increase sharply as the chip power budget grows [7]. 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 [8]. 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 [9]. Skadron et al. [10] proposed the use of control theory algorithms for DTM, using fetch gating and migrating computation as their reaction mechanism. Brooks et al. [8] proposed several localized reactive mechanisms – I-cache throttling, decode throttling and speculation control.

Rohu and Smith [11] 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 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 [12] proposed a *predictive* DTM algorithm targeted at multimedia applications. They intended to show that a predictive combination of architecture adaptation and DVS performs the best across a broad range of applications and thermal limits. Shayesteh et al. [13], Powell et al. [14], and Michaud et al. [15] investigate thread migration via the Operating System as a means of controlling the thermal profile of the chip. They explore the use of swapping applications between multiple cores when a given core exceeds a thermal threshold.

We present the idea of thermal monitoring and response management in a multi-core, multi-threaded environment. The core requirements of this manager are to sense the impact of temperature on various compute structures in a Chip Multiprocessor (CMP), and subsequently respond by reconfiguring the CMP such that the system operates at maximum performance without exceeding its thermal boundary. Pure hardware implementation of thermal management is expensive and lacks scalability in a typical system. On the other hand, a purely software based approach needs instrumentation capabilities to tackle the requirements of managing the low level communication with the CMP cores. Additionally, operating system based implementation lacks flexibility due to strict interface abstractions. These constraints drive us towards proposing a scheme which is minimally invasive to system hardware and software abstraction layers.

III. Thermal Management Architecture

Our proposed architecture has both hardware and software components, as shown in Figure 1. 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. Finally, the hardware platform provides support for processor virtualization features like expanded isolation, and mechanisms for quick thread migration.

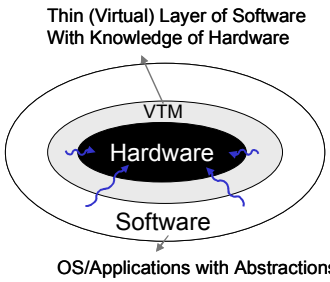

Figure 1. Thermal Manager's System View

The software component of our scheme is the thermal management software that runs natively as a privileged process on the CMP. We assume a thin Virtual Machine Monitor (VMM) running underneath the OS, which is primarily used to enter and exit, as well as pass specific thread information to the Virtual Thermal Manager (VTM) [16]. VTM software resides in

the physical memory that is concealed from all conventional software including the OS. VTM software maintains thermal history tables. Based on the thread specific thermal demands, and the past measurements of the distributed thermal sensors, VTM predicts the thermal mapping for the next epoch of computation. When VTM predicts a potential for thermal hotspots, it takes an intelligent and preemptive measure by reconfiguring the hardware platform. VTM considers the performance and thermal tradeoffs, and provides adjustments for sustained performance levels at target temperature.

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 match the thermal requirements of the CMP. The hardware is assumed to have thermal sensors distributed across the platform. These sensors are assumed to register their readings periodically with memory mapped registers. When VTM is active, it has the highest privileged access to the CMP cores. Once VTM software completes its work to determine and setup the DTM actions, 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. Further details about the VTM framework can be found in [17].

A. Software Details

The main data structures maintained by the VTM software are Thread-to-Core Mapping Table (TCMT) and Temperature History Table (THT). TCMT maintains the thread-to-core mapping of live software threads in the system. The purpose of this table is to keep track of thread mapping, use this information to assist with thread migration and also inform the OS of such actions. TCMT ensures a programmable pre-defined time period between thread migrations, which is fixed at 10ms for this study. This allows enough time for potential thermal imbalance to materialize in our target CMP.

THT constitutes several data structures that are used to predict the future temperature for each distributed sensor in the CMP. An entry is maintained for each live thread, which is responsible for tracking the thread's thermal demands. We used offline profiling of our workloads to initialize these entries classifying each thread as *hot*, *moderate* or *cold*. Alternative techniques to track each thread's thermal behavior at finer granularity and at run-time are a subject of future study. THT also tracks the rate of change of temperature for each sensor in the system. This is accomplished by sampling the thermal sensors at fixed intervals, followed by an update of THT entries with temperature history on VTM invocation. Finally, THT tracks and updates the localized and targeted DTM actions determined by the VTM. The high level software flow for the VTM is presented in Figure 2.

On each invocation, VTM first updates the thermal history in the THT with the current threads to cores mapping, and then starts the process of determining the next threads to cores mapping and possible DTM actions. First, temperature is predicted for all possible threads to cores mapping on the CMP. Details of the temperature prediction are discussed in the next section. Based on the predicted temperature, the final threads to cores mapping is selected based on the following criteria: (1) Largest temperature imbalance exists between two or more cores, and (2) Cores that are not participating in thread migration incur minimum possible performance degradation. This ensures that the CMP will operate at its thermal boundary, while the performance impact of thermal management is minimal.

The predicted temperature delta between thermal sensors on equivalent structures on the logical cores is evaluated for a possible thread migration. If two cores show an imbalance of greater than a pre-defined threshold, thread migration is determined as the possible DTM action. In case when thread migration is evaluated to be infeasible, the VTM software takes a fine-grain approach to thermal management. The future thermal mapping is fed into a lookup based DTM action selection process, which also takes into account the previous action before setting the next action. When the future DTM action is determined for each sensor, a union of these actions is used to setup the knobs for reconfiguring the hardware platform. The next action also appropriately sets the next VTM invocation delay followed by an update to the TCMT and THT tables, and subsequently VTM exits.

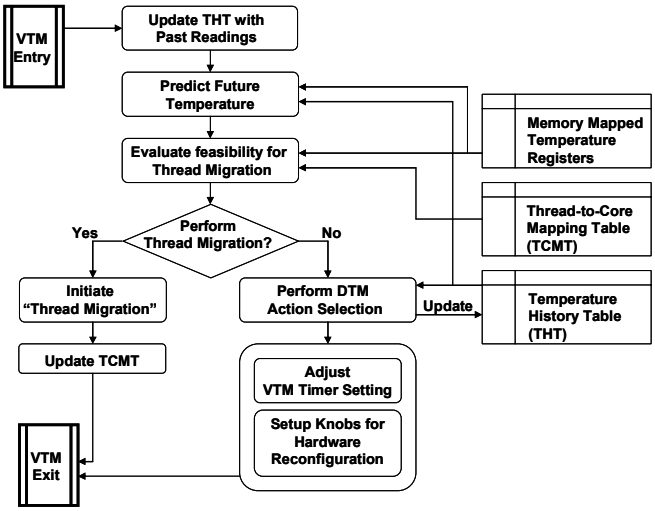

Figure 2. Software Flow

B. Temperature Prediction

Our temperature prediction mechanism relies on past readings to determine the rate of change of temperature. The number of past readings captured by the VTM is controlled by a programmable register and each reading is taken at 100us interval increments before the scheduled VTM invocation. Two temperature readings are selected at a time and the rate of change of temperature is calculated using equation (1):

$$T_{Current} + A.x = T_{Threshold} \quad (1)$$

Where, A is the temperature delta between two selected reading separated by 100us, $T_{Current}$ is the current temperature status of a particular sensor and $T_{Threshold}$ is the temperature threshold for thermal boundary. Finally, x is the unknown variable that determines the number of 100us samples before the temperature is expected to exceed $T_{Threshold}$. We use an averaging function to determine the number of predicted 100us samples.

The temperature prediction is based on the number of predicted 100us samples to exceed thermal boundary and the current DTM actions for the CMP cores. Each DTM action is assumed to lower power density at different rates. This determines the severity of action, which is used to setup the VTM timer for the next invocation. For example, lower severity actions imply that temperature is expected to change slowly; therefore the VTM invocation lag can be large. This allows us to associate a fixed VTM timer delay for each DTM action. Using the predicted samples and the current DTM action, our temperature

predictor associates a discrete rate of change severity level for each thermal sensor. Finally, using the predicted rate of change severity level and the thermal demands of the threads, a final DTM action is selected.

Figure 3 shows a mockup pictorial representation of the temperature prediction mechanism. The red line shows the projected rate of increase based on the past four readings prior to the VTM invocation. The dotted line shows the number of 100us intervals before the projected temperature may exceed the thermal boundary if the DTM actions are not adjusted. Using this information our mechanism predicts future DTM actions such that the temperature stays below the threshold.

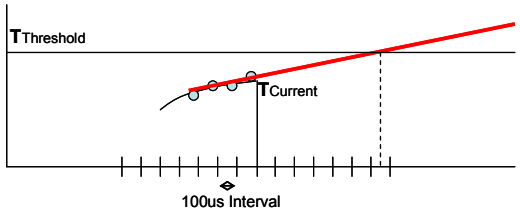

Figure 3. History based Temperature Prediction

C. Thermal Management Action Selection

As described in the previous section, DTM actions are determined based on the rate of increase or decrease of temperature, thread's thermal demands and the last DTM action for the thread to be run next. Our analysis of workloads indicates that a targeted, but gradual increase or decrease in DTM action severity levels yields best results.

In our proposed system we rank DTM actions based on their severity level as shown in Figure 4. 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	VTM Timer Setting
0	1/1	4	4	100%	10 ms
1	1/1	4	4	75%	8 ms
2	1/1	2	2	75%	7 ms
3	1/1	2	2	50%	6 ms
4	1/1	1	1	50%	5 ms
5	1/2	1	1	25%	4 ms
6	1/3	1	1	25%	3 ms
7	1/4	1	1	25%	2 ms
8	1/5	1	1	25%	1 ms
10	Thread Migration				

Figure 4. DTM Actions (Increasing Thermal Severity)

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 [18]. For modeling dynamic power, SESC implements a version of Wattch [19] and Cacti [20] power estimation tools. We have extended SESC to dynamically invoke HotSpot temperature modeling tool [21]. 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.

Core 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

TABLE I. System Parameters

We use quad-core superscalar processor as our example CMP. System parameters used are shown in TABLE I. We assume a low cost cooling package, with maximum tolerable temperature of 85°C. 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 [2], a response latency of 100us, and upper/lower temperature thresholds of 82°C & 81.5°C respectively. Comparatively, we assume that our approach will result in better sensor accuracy, as multiple temperature readings used to project future temperature, statistically provides more accurate readings [10]. VTM is sampled every 1 to 10ms in our setup, with an estimated entry to exit delay penalty of 2000 clock cycles for each invocation. For each thread migration an additional 20us penalty is assumed for flushing core pipeline and transferring the architecture state. We choose 0.5°C temperature threshold to initiate a thread migration, which yields best results in terms of temperature imbalance. VTM temperature threshold is set to 83°C.

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

For each workload, three simulations are conducted. First simulation is without any DTM support and we categorize it as *no-DTM*. Second simulation is with *DVFS* as the DTM response and the trigger is based on the worst case sensor reading. Third simulation is with the VTM framework. Each simulation is run for 1 billion instructions and the performance of each simulation is evaluated based on Millions of Instruction per Second (MIPS). Each simulation is run until the average instruction count seen by the cores reaches 1 billion.

V. Results and Analysis

In this section we analyze the simulation data for a number of multi-threaded workloads running on a quad-core CMP.

A. Analysis of Thermal Behavior

The temperature data is shown for the worst-case unit in each core. Figure 5 shows the thermal behavior for the four threads under DVFS and VTM frameworks. The x-axis shows the relative performance in terms of throughput. Initially, the application runs in full performance mode. But as the thermal sensors observe rising temperatures, VTM proactively adapts to the workload demands and prevent exceeding thermal limits. On the other hand when chip-level DVFS is deployed, a couple of the threads underperform due to the thermal imbalance, thus resulting in an unnecessary performance loss. Although core-level DVFS is possible, the design cost and scalability

implication of such a scheme makes it unfeasible for future technology nodes. As an alternative, VTM provides an architectural solution to manage such workload scenarios. Our VTM approach provides an efficient mechanism to combine thread migration and architectural DTM actions to keep all cores thermally saturated.

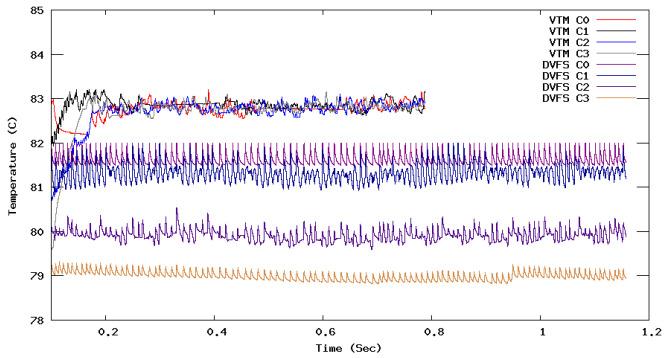

Figure 5. equake-art-gzip-ammp thermal comparisons (VTM vs DVFS)

Figure 6 shows that VTM uses thread migration coupled with fine-grain DTM actions to effectively and consistently manage temperature behavior on the four cores. VTM predicts the future temperature of each thread and takes preemptive actions to keep all cores saturated around the upper temperature threshold. Thread migration is used constantly to keep the imbalance in temperature across all threads to a minimum, while fine-grain DTM actions push for a sustained CMP performance across all cores. As all cores are running threads with unique thermal profiles, higher DTM actions are deployed for the hot threads, while the cold threads run full blown. As soon as the temperature delta between any of the two cores reaches a predetermined differential, the two threads are swapped and DTM actions are adjusted appropriately. The hot and cold threads are regularly swapped around as the cold thread is used to cool down the hot core, while the hot thread heats up the cooled down core. Our results show that thread migration is effective when the cores in the CMP are thermally imbalanced.

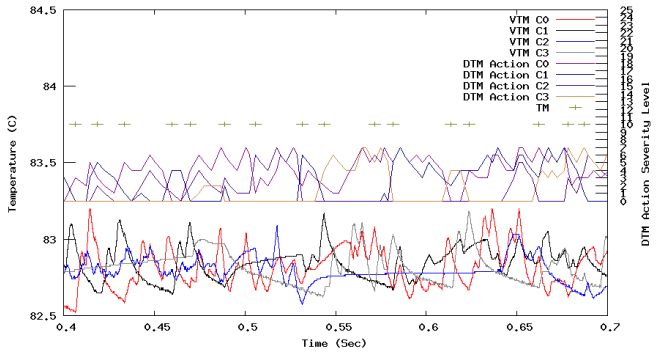

Figure 6. mcf-ammp-gcc-art DTM Actions and Temperature behavior

B. Accuracy of Temperature Prediction

VTM uses a hardware interrupt mechanism to invoke re-evaluation in case of thermal emergencies. A thermal emergency is triggered when a sensor exceeds as a pre-determined temperature. We use this special trigger to evaluate the effectiveness of our temperature prediction mechanism. Figure 7 shows the number of hardware triggers as a percentage of total VTM invocations. Data is shown for all quad-core benchmarks and past history measurements are fixed to four readings. The trigger temperature for hardware interrupt is varied in small increments from 83°C to 83.3°C. As VTM is designed to work at a thermal boundary of 83°C, the hardware interrupt trigger at 83°C incurs 20-30% thermal emergencies. Figure 7 shows that less than 5% hardware interrupts are seen at 83.3°C, thus making our VTM scheme accurate within a 0.3°C temperature limit.

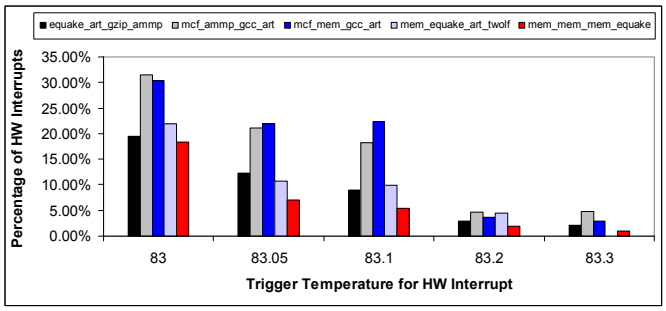

Figure 7. Temperature Prediction Accuracy

We also analyze the impact of varying the length of past temperature measurements in terms of the performance gain/loss and the impact on the hardware interrupts. Figure 8 shows the data for four, six and eight readings compared to a two readings history length. In all cases, the performance impact of varying the history length is negligible (less than ±0.5%). However, the percentage of hardware interrupts shows some significant variance, thus highlighting the impact of history length on the accuracy of predictions. History length of four readings shows the best overall accuracy across all benchmarks. The first two benchmarks show significant increase in hardware interrupts as the number of past reading is increased to eight. This highlights the non-linear behavior of temperature change. As our prediction model averages past readings linearly, it introduces errors. A more accurate model can alternatively use non-linear weights for the past readings to predict the future rate of change.

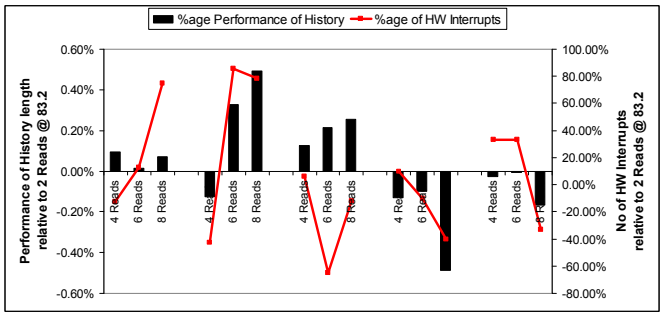

Figure 8. History Length for Temperature Prediction

C. Performance Analysis

Figure 9 summarizes the performance impact of our approach for thermal management compared to DVFS. Our data shows that VTM consistently delivers better performance than the DVFS counterpart. VTM shows an average of 50% improvement over DVFS. DVFS implementation for this study may be considered sub-optimal due to the limited voltage/frequency levels. We consider this fair, as DVS is not predicted to scale for future technology nodes. Additionally, our analysis indicates that global DVFS is more of a problem in terms of performance loss compared to the number of available stepping.

Figure 10 breaks down the execution time into several overheads. Workload is the necessary component consumed by executing the benchmark. All other overheads are related to thermal management. DTM action overhead is the time taken by

DTM actions to degrade performance. The DVFS overhead is the 100us overhead of every DVFS invocation, whereas, VTM overhead is the time taken by VTM to compute next actions. Our data indicates that VTM and DVFS overheads are a negligible percentage of the total execution time. Specifically, VTM overhead plays a minimal role in performance degradation of the processor. This makes the case for using virtual machine as an effective mechanism for thermal management.

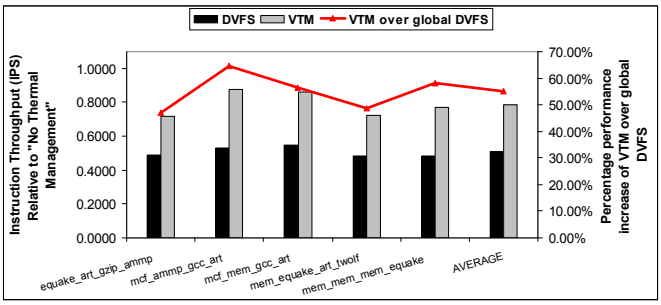

Figure 9. Performance of Thermal Management Schemes

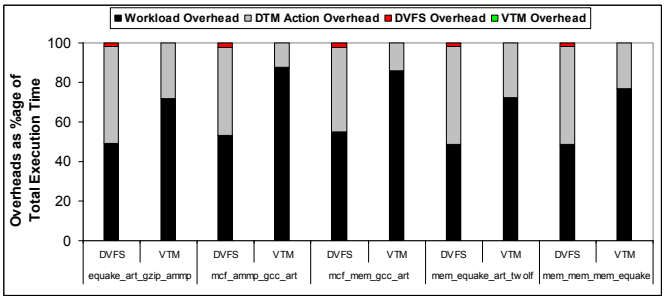

Figure 10. Analysis of thermal management overheads

D. VTM for Reduction of Leakage Power

The proposed use of VTM to keep all cores in the CMP thermally saturated can also be used to reduce the leakage power. Our data shows that some cores when running at elevated temperature with severe hotspots will result in higher leakage, while cooler cores will result in lower leakage. When VTM is deployed, all cores are thermally controlled to stay close to the predetermined temperature levels. Now, the peak temperatures are reduced, but the relatively cooler cores are warmer. Because the leakage power is a nonlinear (exponential) function of temperature, the leakage savings incurred by lowering the hottest temperature outweighs the increase in leakage incurred by increasing the temperature of the cooler cores. Such leakage reductions are conceivable within our proposed architecture, while the rest of the VTM management remains the same. We intend to study such enhancements in our future work.

VI. CONCLUSIONS

We have presented a novel thermal management scheme based on hardware/software co-design 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 a standalone software scheme with hardware assist.

The proposed thermal management scheme adapts to each thread individually to match the computation demands with the hardware thermal profile. The response time is quick because the proposed scheme is not reliant on PLL lock time and decoupling capacitor charge/discharge time. VTM provides a scalable path to the many-core era where DVFS will be limited by physical design and technology constraints. The proposed technique delivers the best performance at target temperature in a distributed thermal sensing environment.

ACKNOWLEDGMENT

This work is supported in part by a grant from National Science Foundation.

REFERENCES

[1] Advanced Configuration and Power Interface (ACPI). Document available at *hhtp://www.acpi.info/spec.html*

[2] S. Naffziger et al., "Power and Temperature Control on a 90nm Itanium®-Family Processor", *Int'l Solid State Circuits Conf, 2005*

[3] "Revision Guide for AMD NPT Family 0Fh Processors", *AMD Publication # 33610, October 2006*

[4] International Roadmap for Semiconductor (ITRS). Document available at *hhtp://public/itrs.net/*

[5] H. Sanchez et. al., "Thermal System Management for high performance PowerPC microprocessors", *COMPCON, 1997*

[6] M. Huang et al., "A Framework for Dynamic Energy Efficiency and Temperature Management", *IEEE Micro, 2000*

[7] S. Borkar, "Design Challenges of Technology Scaling", *IEEE Micro, July-August, 1999*

[8] D. Brooks, M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors", *Int'l Symposium on High Performance Computer Arachitecture, 2001*

[9] T.D. Burd et al., "Dynamic Voltage Scaled Microprocessor System", *IEEE Journal of Solid-State Circuits, Nov. 2000*

[10] K. Skadron et al., "Temperature-aware computer systems: Opportunities and challenges", *IEEE Micro, Nov-Dec. 2003*

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

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

[13] Eren Kursun et al., "Low-Overhead Core Swapping for Thermal Management", *Workshop on Power-Aware Comp. Systems, 2004*

[14] M. Powell, M. Gomma, T. Vijaykumar, "Heat-and-run: Leveraging SMT and CMP to Manage Power Density through the Operating System", *Int'l Conf. on Architectural Support for Programming Languages and Operating Systems, 2004*

[15] P. Michaud et al., "A study of thread migration in temperature-constrained multi-cores", *ACM Transactions on Architecture and Code Optimization, 2007*

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

[17] O. Khan, S. Kundu, "A Framework for Predictive Dynamic Temperature Management of Microprocessor Systems", *Int'l Conf on Computer Aided Design, 2008*

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

[19] D. Brooks et al., "Wattch: A framework for architectural-level power analysis and optimizations", *Int'l Symposium on Computer Arachitecture, 2000*

[20] P. Shivakumar and N. P. Jouppi, "CACTI 3.0: An integrated cache timing, power, and area model", *WRL, Compaq, 2001*

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