Predict and Act: Dynamic Thermal Management for Multi-Core Processors *

Raid Ayoub
CSE Department
University of California, San Diego
La Jolla, CA 92093-0404
rayoub@cs.ucsd.edu

Tajana Rosing
CSE Department
University of California, San Diego
La Jolla, CA 92093-0404
tajana@ucsd.edu

ABSTRACT

In this paper, we propose a proactive dynamic thermal management scheme for chip multiprocessors that run multi-threaded workloads. We introduce a new predictor that utilizes the band-limited property of the temperature frequency spectrum. A big advantage of our predictor is that it does not require the costly training phase like ARMA [7]. Our thermal management scheme incorporates temperature prediction information and runtime workload characterization to perform efficient thermally aware scheduling. Our results show that applying our algorithm considerably improves the average system temperature, hottest core temperature, product MTTF and performance by 6 ^{o}C, 8 ^{o}C, 41% and 72% respectively.

Categories and Subject Descriptors

B.8 [**Performance and reliability**]: General; C.4 [**Computer Systems Organization**]: Performance of Systems

General Terms

Management, Design, Reliability, Performance

Keywords

Temperature Prediction, Thermal Management, Characterization

1. INTRODUCTION

Unprecedented level of technology scaling has magnified the appeal for Chip MultiProcessor (CMP). The increase in the demand for intensive computations and low design cost make CMP a widespread design practice. However, diminution in device geometry coupled with a high level of integration and power dissipation have led to high levels of power density [17]. The primary consequence of such power density is the increase in thermal stress. Unfortunately, dissipating the high temperature requires a large cooling system which increases the cost and the size of the product. Furthermore, the increase in temperature has significant impact on leakage power due to the exponential dependency of leakage power on temperature, an issue that put more pressure on the cooling system. In addition to the cooling challenges, the increase in temperature can result in severe consequences to reliability. Increasing the level of thermal stress can also result in exponential reduction in *mean time to failure, MTTF,* [16]. Additionally, the increase in temperature degrades the performance due to its impact on magnifying the interconnect delay [1]. As the temperature impacts the basic product characteristics, thermal stress is emerging as a stringent constraint in the design of current and future systems.

A number of reactive Dynamic Thermal Management, (*DTM*), techniques have been suggested that throttle the processor activity, e.g. fetch toggling, when the temperature reaches a critical threshold [17]. However the benefit of such class of techniques is limited since they usually cause high performance overhead. Apart from that, such techniques can not tackle the spatial thermal variation in the chip. As the level of integration is steeply increasing, a class of reactive thermal migration techniques have been suggested to improve the temperature balancing across the system resources [8]. However, the benefits of these techniques are limited since they act only when the temperature reaches critical levels. To enhance the temperature management, a few proactive thermal migrations techniques have been suggested recently to prevent the overheating at lower temperature [7, 10, 20]. These techniques are designed based on statistical models, e.g ARMA [7]. Although temperature prediction is fairly accurate, they require a training phase that impacts the performance and prediction opportunities.

In this paper, we propose a cost efficient proactive thermal management scheme that assists in providing thermal aware workload distribution across the processor cores. In order to predict the temperature at minimal cost, we propose a new temperature predictor, called Band-Limited Predictor, BLP, that is based on the *band limited* nature of the temperature frequency spectrum. The concept of the BLP is based on the theoretical work on predicting band limited signals from past samples [13, 11, 14]. Such class of predictors depend primarily on the Nyquist parameter that can be estimated during the design stage. Once the Nyquest parameter is determined, the prediction coefficients can be computed, stored and never recalculated. The analysis that we provide, shows that this predictor is not only accurate but also cost efficient. We incorporate the BLP with thread migration mechanism to minimize the possible overheating and distribute the workload in a thermally sensitive manner. In this work, we are targeting a multicore platform where each core can execute multiple threads. The suggested policy utilizes run time information in terms of fetch rate to determine the appropriate

*This work has been funded in part by Sun Microsystems, UC MICRO, Center for Networked Systems (CNS) at UCSD, MARCO/DARPA Gigascale Systems Research Center and NSF Greenlight.

ISLPED'09, August 19–21, 2009, San Francisco, California, USA.

thread that should be migrated among the set of threads currently assigned to the hot core. The results that we provide prove the applicability of the suggested approach.

The rest of the paper is organized as follows: The related work is given in section 2. Section 3 is dedicated to describing our proactive thermal management and the details of the suggested predictor. In section 4 the evaluation methodology and the experimental results are discussed. The conclusions are provided in section 5.

2. RELATED WORK

In recent years, number of *DTM* techniques have been suggested to tackle the overheating problem in processors. The research in [4] has proposed two reactive *DTM* techniques that are able to manage aggressive heat. The first technique is based on voltage-frequency scaling while the other prevents the fetch on a regular basis. Recently, *DTM* techniques have become an integral part of actual processors. For example, the Pentium 4 employs a clock gating approach to manage the overheating problem [9]. The research in [8, 6] proposes thermal activity migration to manage the excess in temperature through migrating the computations across duplicated units. The common drawback in these approaches are the non-trivial performance overhead and poor thermal distribution across the die. To overcome these problems a class of proactive thermal management techniques have been suggested to prevent the overheating at lower temperature. The authors in [7] propose the ARMA model that is based on the serial autocorrelation in the temperature time series data. The model is updated dynamically to adapt to possible workload changes. Although the ARMA model is fairly accurate, it requires a training phase that could impact the performance, and the predictor could miss some of the prediction opportunities during training. As executing ARMA during training mode generates additional heat that may alter the processor temperature footprint, such scenario could lead to temperature misspredictions. The authors in [10, 20] suggest proactive thermal management techniques that utilize regression-based thermal predictors. However, these techniques also suffer from run-time adaptation overhead.

To the best of our knowledge, there exist no prior work on proactive dynamic thermal management that requires no predictor training. The primary contributions of our work can be summarized as: 1) Introducing a new thermal predictor that is based on the band limited property of the temperature spectrum which is workload independent. 2) We outline and resolve the challenges of extracting the heat impact of individual threads in a multithreaded environment. 3) We propose a proactive thermal management policy for managing the overheating in a multicore environment where each core can run multiple threads. 4) We present a thorough evaluation and discussion of the proposed technique with benefits and overhead.

3. PREDICTIVE THERMAL MANAGEMENT

In this work we are targeting a multicore platform where each core can execute multiple threads e.g. SMT core. Figure 1 depicts the operational framework of the proposed approach. The OS scheduler employs proactive thermal management to minimize the impact of heating problem. To predict the core temperature efficiently we propose a fundamentally new temperature predictor that is based on the concept of predicting band-limited signals which requires no training phase. The predictor coefficients are estimated at the design time based on the temperature spectral limited bandwidth. The moving history of the temperature is fed to the suggested predictor to estimate the future temperature, $T_0 + \Delta T$ where

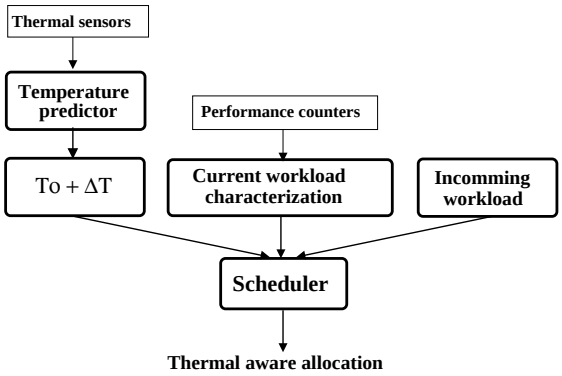

Figure 1: Thermal aware scheduling

T_0 represents the current temperature. The temperature is collected using core's thermal sensors that are commonly available in the state-of-the-art processors and can be accessed easily by the operating system. The workload characterization (Figure 1) is done to estimate the contribution of the individual threads to the core heating. To extract the activity of the individual threads we use performance counters that are typically available in current processors. The last input to the OS scheduler are the queues that contain the tasks that are waiting to be executed (incoming workload).

In the proposed framework, the OS schedular calculates the predicted temperature of the individual cores at each scheduling period. In case that the predicted temperature of one of the cores surpasses a given temperature threshold, the schedular migrates portion or all of the workload of the hot core to the core that is predicted to be the coldest. Preventing the overheating at lower temperature also reduces the impact of the exponential dependency of leakage power on temperature in inducing additional heat. The migration overhead is only a few microseconds which is much lower than the cost of putting the processor in a low power mode for cooling [3, 12, 6]. The migration cost comes primarily from the operating system and migrating the thread state [12]. The workload characterization is used to improve the efficiency of migrations in the cases when only portion of the running threads are allowed to be migrated. In handling the incoming threads, the schedular assigns them to the core that is predicted to be the coldest. In the subsequent sections we elaborate on the design of our suggested approach.

3.1 Temperature predictor based on temperature band limited spectrum

The concept of the suggested predictor in this paper is based on the fact that band limited signals can be predicted from the past samples using prediction coefficients that are *independent* of the particular signal samples or autocorrelation function [13, 11, 14]. In such class of predictors the prediction coefficients are function of the signal bandwidth only. Before we elaborate on our band limited predictor, BLP, we show that the temperature spectrum satisfies the band limited condition. The temperature is typically modeled as an RC network [17]. The die individual components temperature are modeled as *low pass RC* filters with lateral resistances connecting the adjacent units. This indicates that the temperature at each unit constitutes a low pass filter. The values of the lateral resistances are typically high, which can be neglected to simplify the analysis [8]. As a result, the temperature spectrum at each component can be computed as follows:

$$\frac{Temp(w)}{Temp(0)} = \frac{1}{\sqrt{1 + (w\tau)^2}} \quad (1)$$

where $Temp(w)$ represents the temperature value as a function of the angular frequency w, and τ is the temperature time constant that equals to RC that can be extracted at the design time using die thickness and physical characteristics of the thermal package. The value of the die τ is approximately the same across the die components. As the temperature represent a low pass filter, it can be concluded that the temperature frequency spectrum satisfies the band limited condition. The work in [11] show that band limited signals can be predicted using the following linear formula:

$$x(t) = \sum_{n=1}^{N} a_n x(t_n) \tag{2}$$

where a_n are the prediction coefficients, t_n represents the nth time sample and N is the number of total samples. For the case of uniform sampling, the value of t_n can be written as $t_n = t - nT_s$ where T_s is the sampling period. For this predictor to be applicable, the error needs to be bounded and sufficiently small. The absolute error is expressed by $\epsilon = |x(t) - \sum_{n=1}^{N} a_n x(t_n)|$. Using Paley-Wiener theorem and Schwarz formula, a bound on the error ϵ^2 can be found as follows [11]:

$$\epsilon^2 \leq \{\int_{-W}^{W} |X(f)|^2 df\}\{\int_{-W}^{W} |ds(f)|^2 df\} \tag{3}$$

where $ds(f) = e^{i2\pi ft} - \sum_{n=1}^{N} a_n e^{i2\pi f t_n}$, f is the frequency and W is the highest frequency in Hertz. The first part of this equation represents the signal energy while the second part is the amount of prediction error, that can be written as:

$$\epsilon^2 \leq \|x\|^2 \cdot \epsilon_I \tag{4}$$

where ϵ_I is the error part while $\|x\|$ is the energy. The error integral represents an *N-dimensional* function of prediction coefficients. The BLP coefficients can be obtained by minimizing the error part in (3). The optimal coefficients can be extracted using standard method of minimization, [14], that gives the following system of equations:

$$\sum_{n=1}^{N} a_n sinc(2W(t_j - t_n)) = sinc(2W(t_0 - t_j)) \tag{5}$$

where j = 1,...,N and $sinc(t) = sin(\pi t)/(\pi t)$. If we label the system matrix D and the right side vector as **b**, then the prediction coefficients can be extracted through solving the system:

$$D\mathbf{a} = \mathbf{b} \tag{6}$$

where **a** is the prediction coefficients vector of $N \times 1$ size, **b** is of $N \times 1$ size and can be written as $\mathbf{b} = [s(t_0 - t_1), s(t_0 - t_2), s(t_0 - t_3), ..., s(t_0 - t_N)]^T$ and D is $N \times N$ matrix that can be expressed as:

$$D = \begin{pmatrix} s(t_1 - t_1) & s(t_1 - t_2) & s(t_1 - t_3) & ... & s(t_1 - t_N) \\ s(t_2 - t_1) & s(t_2 - t_2) & s(t_2 - t_3) & ... & s(t_2 - t_N) \\ s(t_3 - t_1) & s(t_3 - t_2) & s(t_3 - t_3) & ... & s(t_3 - t_N) \\ ... & ... & ... & ... & ... \\ s(t_N - t_1) & s(t_N - t_2) & s(t_N - t_3) & ... & s(t_N - t_N) \end{pmatrix}$$

where $s(t) = sinc(2Wt)$. It can be seen that the optimal prediction coefficients depend only on the signal bandwidth, W. As the temperature frequency bandwidth depends on the temperature time constant, it can be concluded that the temperature predictor coefficients can be determined at the design time. For the case of uniform sampling, the upper bound for prediction distance, T_s, can be obtained using Nyquist condition, $2T_sW < 1$. The results in [14] show that this predictor is highly accurate. To estimate the predictor coefficients for our platform we only need the temperature spectral bandwidth which depends on the die τ as shown in (1). Extracting τ can be simply accomplished at the design time using die layout and standard thermal parameters [17].

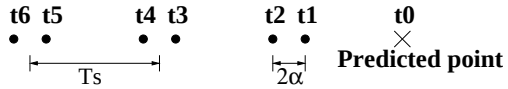

Figure 2: Position of interlaced sampling points

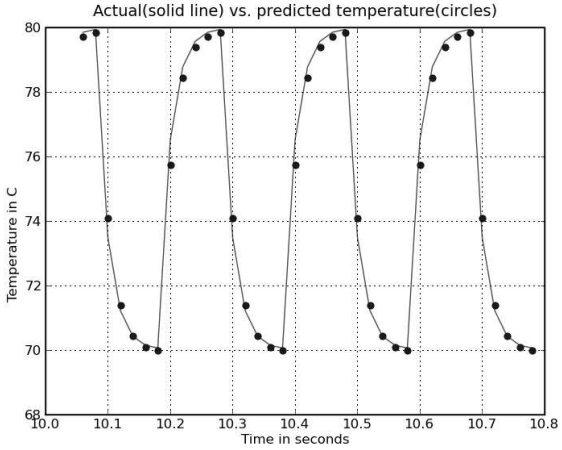

Figure 3: Temperature prediction using BLP

The temperature typically changes slowly as a function of its thermal time constant. As a result, extending the prediction window allows for better thermal management as more overheating could be captured and then prevented at an earlier point. Interestingly, the prediction window can be extended through employing nonuniform sampling. The theoretical work in [15] shows that if a signal is sampled at $1/m$ times the Nyquist rate, but in each sampling interval not one but m samples are used, the signal can be reconstructed completely. In [11], the author show that this concept can be extended to increase the prediction window. To apply this concept, the samples need to be placed in interlaced pattern. For $m = 2$, the placement of samples should follow a pattern of two close samples that is followed by a uniform delay, then two more close samples are taken and so on. Figure 2, depicts the positions of interlaced sampling points. The theoretical proof is given in [11]. The prediction distance is computed as $T_s - 2\alpha(m-1)$ where α is the half distance between the two close samplings.

In order for the predictor to be feasible, we need to show that the prediction window is in the range of operating system scheduling period which is typically 10ms. Temperature spectrum bandwidth depends on its time constant which is in the range of 20ms for a representative die thickness of 0.2mm with thermal package characteristics typical for designs with spreader and heat sink as in [17]. Using the analysis that we have provided, the feasible parameters that allows for a prediction distance to be around the range of OS scheduling period are: $m = 3$, α=$(0.12 - 0.14)T_s$, N (3-6), and sampling time (T_s) is in the range of τ. The values of α is chosen as a tradeoff between prediction distance and accuracy. This is because smaller values of α lead to larger coefficients that make the predictor more prone to noise. In Figure 3, we show an illustrative example of applying the BLP to a temperature signal. In this example we assume τ = 20ms, $T_s = \tau$, $\alpha = 0.125T_s$. It is clear from the figure that BLP is fairly accurate (error ≈ 0.2 ^{o}C) even though there are large variations in the temperature signal. Apart from the good accuracy, the prediction overhead is minuscule, as generating new prediction requires computing a simple linear polynomial. The cost of computing new prediction is negligible as it takes only several CPU cycles.

3.2 Thermal aware thread scheduling

In this section we focus on developing an algorithm for distributing the workload across the available cores in a thermal sensitive manner. The primary objective is to minimize the frequency of hot spots, lower the overall system temperature, and the thermal spatial variations across the cores. The proposed scheduler utilizes our BLP to perform proactive job scheduling.

Current operating systems employ **dynamic load balancing, DLB**, to enhance the utilization of the system resources. The DLB performs thread migration to shrink the difference in task queue length of the individual cores [6]. The operating system initiates dynamic load balancing every hundred milliseconds. Employing such policy, however, is not efficient in mitigating the hot spots. This is because the DLB does not consider the temperature metric in allocating the workload. When the number of running threads is less than the number of physical cores, the DLB does not initiate any migrations since the workload is balanced from the DLB point of view. However, such scenarios could result in hot-spots as portion of the cores could be highly active while the others are idle. In case when the total number of active threads exceeds the number of physical cores but lower then the number of hardware threads (logical processors), the DLB can minimize the differences in task queue length but may still cause frequent hot-spots as it might cluster hot threads together on the same core. Additionally, the performance can be degraded as frequent overheating results in putting the cores in a low power mode more often. In this work we implement the DLB as our *base line policy* for the purpose of comparison. In the following discussion we elaborate on the other polices that we have implemented.

Reactive Thermal Management, RTM: Thermal aware scheduling can be attained at run time through adjusting the workload of the hot cores. One common approach is to migrate the threads from the cores that reache the migration threshold to a colder ones if possible [8, 12]. Upon reaching the migration threshold, the RTM selects randomly one of the threads and migrates it away. To avoid possible ping-ponging, we initiate the migration process only when there is a sufficient temperature difference between the hot cores and the colder ones.

Proactive Thermal Management using ARMA, PTM-ARMA: We also implemented another proactive policy that utilizes the ARMA based temperature predictor proposed in [7] for the comparison purpose. Our implementation, PTM-ARMA, uses ARMA to predict the future temperature. If the predicted temperature is higher than the migration threshold, the schedular selects randomly one of the threads from the hot core and migrates it to the core that is predicted to be the coldest and has available resources. For forming the ARMA(p,q) model, we used the iterative approach as described in [7] and set the maximum order for p and q to 5 (p and q represent the order of the autoregressive and moving average part of the model respectively). At run time, we check if the prediction error exceeds 5 ^{o}C, and if it does we initiate adaptation phase and form a new model. To avoid noncritical adaptations, the error is calculated by averaging the prediction errors in a window of 20 samples. During the adaptation phase, the ARMA thread is allowed to execute only if there is an available hardware threads.

Proactive Thermal Management using BLP, PTM-BLP: Our proactive policy performs thermal aware scheduling through either migrating the hot threads across the system cores or scheduling the incoming jobs in a thermally aware manner. To initiate the migration process, the predicted temperatures of the hot cores must surpass the migration threshold and there has to be a sufficient difference in the predicted temperature between the hot and coldest cores. When the coldest core is idle, we migrate the most active thread from the hot core to the colder one. If the coldest core is executing, we select the thread with modest activity from the hot core and migrate it to balance the temperature.

Table 1: Simulation parameters

Parameter	Value
Issue width	4
Number of threads	2
ROB	128
Functional units	4 IntALU, 1 IntMult/Div 1 FPALU, 1 FPMult/Div
Branch predictor	Tournament 2048 local predictor 8192 global predictor
BTB	2K entries, 1 way
LSQ	32
L1 I-cache	32KB, 4 ways, 32B blocks, 1 cycle
L1 D-cache	32KB, 4 ways, 32B blocks, 1 cycle
L2	4MB, 8 ways, 64B blocks, 12 cycles
Memory latency	200 cycles

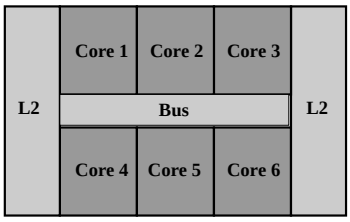

Figure 4: Processor floorplan

To extract the activity of the individual threads, we use *fetch rate* as a run time workload characterization mechanism. The fetch rate can be measured directly through the use of processor performance counters (e.g. Model Specific Registers MSR). The fetch rate statistics are collected for the duration that equals to one OS scheduling period. When assigning the incoming jobs, we use the result of prediction to find the coldest cores with available resources [7].

4. EVALUATION

4.1 Methodology

For the experimental evaluation we assume a platform that consists of 6 cores, where each core is a 4-issue SMT ALPHA like processor that can run at most 2 threads. Table 1 gives the simulation parameters that we have used in our simulations. We utilized three simulators, M5 [2], Wattch [5], and HotSpot-4.0 [17]. The M5 simulator is used to obtain the architectural level performance simulation. We further extend M5 for incorporating the discussed policies in section 3.2 in the OS scheduler. The M5 results are fed into Wattch to obtain the power values of the processor functional units. The power values are then used to estimate the temperature through the HotSpot simulator. Figure 4 shows the floorplan obtained by scaling ALPHA 21264 processor into 65nm technology. In our simulations we use processor clock speed of 2GHz. L2 cache area is estimated based on Cacti simulation tool [19]. The die thickness is 0.2mm and the thermal package is similar to the one in [17]. To account for CPU cores leakage power temperature dependency, we used the second-order polynomial model that is proposed in [18]. We extracted the model coefficients empirically based on the given normalized leakage values. To estimate the leakage values for 65nm technology we incorporate the reported value of leakage power density ($0.5W/mm^2$ at 383K [3]) in the second-order polynomial model. For the crossbar bus power consumption, we scaled the power depending on the activity in the cores and L2 cache. In our results we also account for the temperature warmup, as the heat sink has a longer time constant than the die.

Table 2: Benchmarks characteristics

Benchmark	Avg. temperature ^{o}C	Temp. Standard deviation	Over heating(%)	Dcache access(%)
gcc	68.63	5.80	1.61	13.79
gzip	75.14	2.03	2.06	17.58
swim	78.47	5.00	16.41	8.76
bzip2	86.14	5.01	60.07	29.18
crafty	91.99	0.51	99.12	43.80

Table 3: Benchmark combination list

Test group	Benchmarks
1	4× bzip2
2	5× bzip2
3	3× bzip2 + 3×gzip
4	3× crafty + 3× swim
5	3× bzip2 + 3× gzip + 1× gcc
6	3× crafty + 3× swim + 1× gcc
7	3× bzip2 + 3× gzip + 2× gcc
8	3× bzip2 + 3× gzip + 3× gcc

We use benchmarks from the SPEC2000 suite for our workloads (see Table 2). We selected a set of benchmarks that exhibit various levels of thermal stress to represent real life applications. We ran each benchmark for a representative interval of 4 seconds. Such time is sufficient to evaluate our policies since it is orders of magnitude larger than the die thermal time constant ($\sim$ 20ms). As we are assuming an SMT multicore platform we constructed a representative set of benchmarks that are shown in Table 3. The selected benchmark combinations varies in terms of CPU utilization and thermal stress, to better evaluate our policies under various workload conditions.

We assume the critical temperature as 85 ^{o}C which is common to such designs [7]. The migration is allowed to be initiated only when the temperature difference between the hot core and colder core is 5 ^{o}C to avoid a ping-pong effect. For the ambient temperature we assume 45 ^{o}C. For the predictor we use a conservative prediction distance of 8.5ms to account for possible noise in reality. However, our predictor is capable of providing appreciable prediction with higher prediction distances as it is shown in section 3.1. We assume $\alpha = 0.135$, $m = 3$ and N=3 based on the analysis in section 3.1. The temperature sampling is obtained using the method that is discussed in section 3.1. In case the thermal management policy is unable to eliminate the overheating, we use a back up DTM that puts the hot cores in a low power mode for 100 μs and repeats this process as necessary.

4.2 Results

4.2.1 Thermal Improvement

The reported results show that our policy outperforms the other policies in minimizing the system average temperature as well as the temperature of the hot cores. Minimizing these metrics has a big advantage on lowering the overall cooling costs as well as improving the product MTTF.

Figure 5 shows the die *average temperature* when using the DLB, RTM, PTM-ARMA, and PTM-BLP policies for all the workloads given in Table 3. It is evident from the results that using PTM-BLP results in a considerable improvement over the DLB, RTM and PTM-ARMA; the enhancement is as high as 8.8 ^{o}C, 8.4 ^{o}C, and 8.16 ^{o}C respectively. It provides appreciable savings over the DLB since it is able to prevent the overheating at lower temperatures and

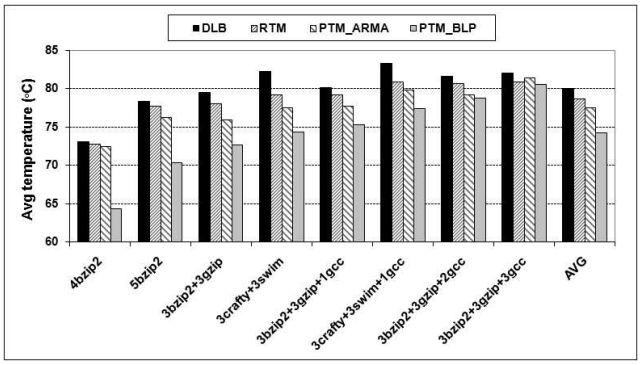

Figure 5: Lowering system average temperature

distribute the heat more evenly across the cores. The additional source of savings comes from lowering the impact of exponential dependency of leakage power on temperature since preventing the overheating at earlier point reduces the effect of the dependency positive loop. As can be seen from the results, the savings in the case of $4bzip2$ and $5bzip2$ are higher than $\{3bzip2 + 3gzip + 2gcc\}$ since there are more opportunities for migrating the threads across the system in the prior cases. Interestingly, our scheme is capable of providing savings even when the number of threads exceeds the number of cores, (e.g. $\{3bzip2 + 3gzip + 1gcc\}$). The PTM-ARMA outperforms the reactive policy (RTM) since the RTM acts only when the temperature becomes close to the overheating point. More interestingly, our scheme shows considerable improvement even over the proactive policy, PTM-ARMA. This could be attributed to the impact of prediction inaccuracy during training phases as the PTM-ARMA defaults to its current model until the training phase is complete during such events. It should be noted that training phase could be a frequent event. In addition, the ARMA training phase can take up to several hundred milliseconds [7], which is expected to increase the overall power consumption and the temperature in turn. Beside that, periods with frequent thread migrations can cause the temperature dynamics to vary and reduce the prediction accuracy. The other factor that contributes to the reduction in the average temperature is the use of runtime workload characterization that identifies activity of the individual threads and allocates them in a thermally sensitive manner.

Figure 6 shows the benefit of applying our scheme to minimizing the temperature level of the hottest cores in the system. It can be seen from the results that using PTM-BLP results in a strong improvement over the DLB, RTM and PTM-ARMA; the improvement is as high as 13.3 ^{o}C, 7.8 ^{o}C, and 6.37 ^{o}C respectively. The strong reduction in the hottest core temperature reflects a great tem-

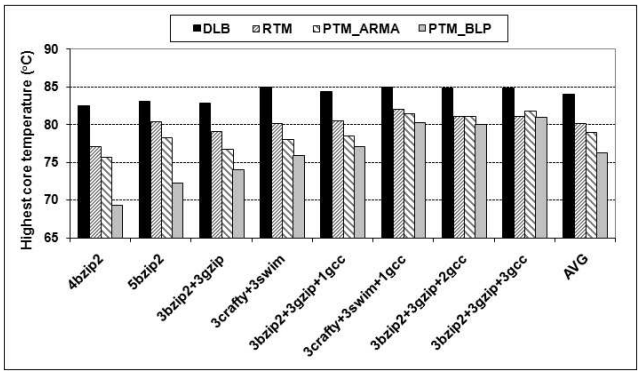

Figure 6: Minimizing cores highest temperature

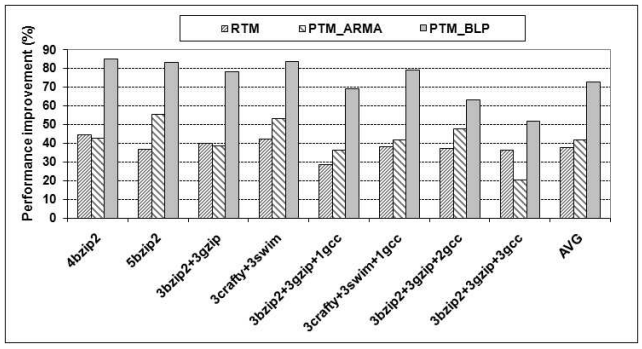

Figure 7: Performance improvement over DLB

perature prediction accuracy of our method. The maximum benefit occurs in the case of $4bzip2$ since it has the highest opportunity of finding available hardware threads. For the case when the number of running threads is similar, the improvement of using our scheme over the RTM and PTM-ARMA is shown to be higher in the case of $\{3bzip2+3gzip\}$. This can be related to the higher temperature average of $\{3crafty+3swim\}$ which lowers the chance of passing the temperature difference threshold that is required to avoid the ping-pong effect. Our scheme outperforms the PTM-ARMA by 2.9 ^{o}C degrees on average.

4.2.2 Performance Impact

Figure 7 illustrates the percentage of performance improvement over the DLB when using the DTM policies that are discussed earlier. Our policy significantly outperforms DLB, RTM and PTM-ARMA; the enhancement is as high as 84%, 72% and 74% respectively. The DLB delivers the worst performance as it suffers the most from powering down overhead due to the frequent overheating. PTM-BLP surpasses RTM appreciably since the RTM could lead to more frequent power down events since the average temperature in the case of the RTM is higher than PTM-BLP (see Figure 5). The other reason is due to many migrations that are not helpful since RTM makes thermal management decisions based only on the current temperature. The results show that applying PTM-ARMA could result in high performance overhead that exceeds the level of the RTM. Example of that can be seen in the case of the workload $\{3bzip2+3gzip+3gcc\}$. This is due to the training overhead and resulting inaccuracy due to frequent migrations. Including workload characterization in the schedular as the PTM-BLP does, assists in improving the performance due to the enhancement in migration efficiency. Interestingly, our scheme outperforms the other techniques even in the case of high utilization. These results indicate that PTM-BLP is highly effective in handling overheating at various levels of system utilization.

4.3 Conclusion

In this work, we have introduced a new proactive dynamic thermal management technique for multicore system. Our algorithm incorporates continuous temperature prediction information and runtime workload characterization to guide the OS schedular in allocating the workload in a thermally sensitive manner. We also introduced a new temperature predictor that not only requires no runtime adaptation, but is also highly cost efficient. We provide detailed analysis on how to calculate the prediction coefficients at the design time. We implemented our scheme and compared it against other state-of-the-art reactive and proactive polices. Our results show that applying our algorithm significantly improves the average system temperature, hottest cores temperature and performance by 6 ^{o}C, 8 ^{o}C and 72% respectively. The amount of reduction in average temperature is exponentially related to MTTF, so original MTTF of 5 years, now becomes 8.6 years.

5. REFERENCES

[1] A. Ajami, K. Banerjee, and M. Pedram. Modeling and analysis of nonuniform substrate temperature effects on global interconnects. *IEEE Trans. on CAD*, pages 849–861, 2005.

[2] N. Binkert, R. Dreslinski, L. Hsu, K. Lim, A. Saidi, and S. Reinhardt. The m5 simulator: Modeling networked systems. *IEEE Micro*, 26(4):52–60, 2006.

[3] P. Bose. Power-efcient microarchitectural choices at the early design stage. *Workshop on Power-Aware Computer Systems*, 2003.

[4] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. *HPCA*, pages 171–182, 2001.

[5] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. *ISCA*, pages 83–94, 2000.

[6] J. Choi, C. Cher, H. Franke, H. H. A., Weger, and P. Bose. Thermal-aware task scheduling at the system software level. *ISLPED*, pages 213–218, 2007.

[7] A. Coskun, T. Rosing, and K. Gross. Proactive temperature management in MPSOC. *ISLPED*, pages 165–170, 2008.

[8] S. Heo, K. Barr, and K. Asanovic. Reducing power density through activity migration. *ISLPED*, pages 217–222, 2003.

[9] G. Hinton, D. Sagar, M. Upton, D. Boggs, D. Carmean, and P. Roussel. The microarchitecture of the pentium 4 processor. *Intel Technology Journal*, 5(1):1–12, 2001.

[10] A. Kumar, L. Shang, L. Peh, and N. Jha. Hybdtm: A coordinated hardware-software approach for dynamic thermal management. *DAC*, pages 548–553, 2006.

[11] F. Marvasti. *Nonuniform Sampling Theory and Practice*. Kluwer Acedemic/Plenum Publishers, New York, 2001.

[12] G. Mohamed, M. Powell, and T. Vijaykumar. Heat-and-run: leveraging smt and cmp to manage power density through the operating system. *SIGOPS Oper. Syst. Rev.*, 38(5):260–270, 2004.

[13] D. Mugler. Computationally efficient linear prediction from past samples of a band-limited signal and its derivative. *IEEE Transactions on Information theory*, 36:589–596, 1990.

[14] D. Mugler. Computational aspects of an optimal linear prediction formula for band-limited signals. *Computational and applied mathematics*, pages 351–356, 1992.

[15] A. Papoulis. *Signal analysis*. McBraw-Hill, London, 1981.

[16] M. Pedram and S. Nazarian. Thermal modeling, analysis, and management in vlsi circuits: principles and methods. pages 1487–1501, 2006.

[17] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. *ISCA*, pages 2–13, 2003.

[18] H. Su, F. Liu, A. Devgan, E. Acar, and S. Nassif. Full chip leakage estimation considering power supply and temperature variations. pages 78–83, 2003.

[19] D. Tarjan, S. Thoziyoor, and N. Jouppi. Cacti 4.0. *Technical report, HP Laboratories*, pages 1–15, 2006.

[20] I. Yeo, C. Liu, and E. Kim. Predictive dynamic thermal management for multicore systems. *DAC*, pages 734–739, 2008.