An Adaptive Temperature Threshold Schema for Dynamic Thermal Management of Multi-Core Processors

Bagher Salami
Computer Engineering Department
Ferdowsi University of Mashhad
Mashhad, Iran
bagher.salami@stu-mail.um.ac.ir

Mohammadreza Baharani
School of ECE
University of Tehran
Tehran, Iran
m.baharani@ut.ac.ir

Hamid Noori
Computer Engineering Department
Ferdowsi University of Mashhad
Mashhad, Iran
hnoori@um.ac.ir

Abstract - This paper presents an adaptive task migration threshold schema for dynamic thermal management of multi-core processors to minimize both average and peak temperature with very low performance overhead. Our proposed algorithm adjusts temperature threshold according to processor work-load and hardware platforms. The experimental results indicate that our technique can significantly decrease average and peak temperature compared to Linux standard scheduler, and two well-known thermal management techniques.

Keywords— dynamic thermal management; multi-core system; temperature threshold; DVFS; task migration

I. INTRODUCTION

As feature size is shrinking, ability to implement complex Chip Multiprocessors (CMPs) with larger number of cores increases. However, due to increased density and complexity of CMPs, the power consumption and generated temperature of modern processors is increasing that threatens system reliability [1]. Dynamic Thermal Management (DTM) techniques are proposed to mitigate the aforementioned problem. One of DTM techniques is task migration that moves tasks from hot cores to cool cores in order to control and manage the overall temperature. Temperature threshold of starting task migration in all prior proposed task migration algorithms is fixed [2-3]. The temperature threshold of starting task migration can be affected dramatically by dynamic behavior of runtime workloads and different physical behaviors of hardware platforms from one CMP to another. Motivated by this fact, an Adaptive temperature Threshold schema for Dynamic Thermal Management (ATDTM) is proposed. ATDTM uses cores temperature and migration frequency to adjust temperature threshold. The experimental results indicate that the proposed algorithm lowers average and peak temperature with negligible performance overhead compared to Linux standard scheduler, TAS [2], and PDTM [3].

II. PROPOSED ALGORITHM

The main parts of algorithm are: *Threshold Adjustment*, *Tasks Assignment*, and *DVFS Adjustment*. In *threshold adjustment*, the value of temperature threshold (T_{thr}) is tuned regard to both migration frequency ($Migration_{\#}$) and migration limitation ($Migration_{limit}$). If the total number of migrations in the last ten iterations is higher than $Migration_{limit}$, T_{thr} is increased and if the total number of migrations in the last ten iterations is zero, T_{thr} is decreased. Note that the T_{thr} cannot be more than critical temperature of processor. The higher migration frequency causes more performance degradation. Therefore, our proposed threshold management tries to control migration frequency and prevent it from increasing. Note that the higher temperature threshold causes the migration frequency to decrease. However, the increase of both temperature and task migration degrade the overall system performance. Our proposed *Threshold Adjustment* tries to find a trade-off between temperature threshold and task migration. *Task assignment* is activated when there is at least one core in the critical situation. A core is in critical situation when it reaches to T_{thr} in less than t_{res}, where t_{res} is the response time for the algorithm to take an action to decrease the core temperature. In this situation, the algorithm finds the appropriate core according to t_r and t_{res}, where t_r is the predicted time when the $core_i$ reaches to T_{thr}. Our prediction method is modified version of [2]. If the algorithm cannot find a suitable destination core, it decreases the core frequency (f_{cur}) to decrease the temperature. In *DVFS adjustment*, if there is not any migration and current frequency is lower than predefined minimum frequency (f_{min}), the algorithm increase frequency to improve performance. The ATDTM algorithm is as follows:

Algorithm 1 ATDTM scheduler algorithm

```
1: If migration# > migrationlimit then increment(Tthr)
2:    else If migration# == 0 then decrement(Tthr)
3:         endif
4: endif
5: for i =1 to Core_count do
6:    Predict tr [i] for corei
7: endfor
8: while (there is at least one corei that tr [i] <= tres) do
9:    if there is any core that tr [i]>= tres then
10:      Migrate task from critical cores to appropriate core.
11:      else  decrement(fcur)
12:   endif
13: endwhile
14: if fcur <=fmin && Migration# == 0  then increment(fcur)
15: endif
```

III. EXPERIMENTAL RESULTS

This section provides experimental results for five different applications from SPEC CPU2006 benchmarks: hmmer, gcc, libquantum, gromacs, and bzip2. we use Intel Core i7-2600 processor. The kernel version of Linux is 3.2.0 and the size of system main memory is 8GB. The LM sensor [4] application is used to read cores temperature. We also use cpufreq tool to adjust processor frequency. In all of our experiments the fan speed has been fixed to a constant RPM (Rotation per Minutes). The value of t_{res}, $migration_{\#}$ and f_{min} are set to 2 seconds, 5, and 2GHz respectively.

Fig. 1 illustrates cores temperatures for TAS, PDTM, Linux standard scheduler, and our proposed algorithm by running five programs simultaneously. The temperatures are sampled every second. Our proposed ATDTM reduces average temperature about 13.4%, and reduces peak temperature about 13.3% with 3.6% performance overhead compared to standard Linux scheduler. The experimental results also indicate that our proposed algorithm reduces average temperature about 1.7% and 10.2% compared to TAS and PDTM, respectively. ATDTM reduces peak temperature about 6.7% compared to both TAS and PTDM. The overall system performance overhead is about only 0.8% compared to PDTM and its performance overhead is same as TAS. Table I summarizes the comparison results for these four algorithms.

TABLE I
Comparison results of ATDTM against Linux, TAS, and PDTM.

DTM Algorithm	Average Temp.	Peak Temp.	Run Time(Sec)
Linux	58.2(°C)	68(°C)	612(Sec)
PDTM	56.5(°C)	64(°C)	630(Sec)
TAS	52.2(°C)	64(°C)	635(Sec)
ATDTM	51.3(°C)	60(°C)	635(Sec)
Improvement ATDTM vs. PDTM	10.2%	6.7%	-0.8%
Improvement ATDTM vs. TAS	1.7%	6.7%	0%
Improvement ATDTM vs. Linux	13.4%	13.3%	-3.6%

In the Linux scheduler, one core starts program execution and terminates it due to high temperature threshold assignment. Therefore, it does not use other cooler cores to decrease the hot core temperature. PDTM tries to mitigate problem using core temperature prediction; however, it cannot find a proper core when all cores temperatures are near to temperature threshold (70°C). In such circumstances, the task migrates between different cores repeatedly. This phenomenon is known as Ping-Pong effect [1]. TAS categorizes applications based on their thermal behavior to improve prediction accuracy. It also defines an appropriate core as a core which reaches to temperature threshold late. Our proposed technique improves TAS algorithm using an adjustable threshold schema.

IV. CONCLUSION AND FUTURE WORKS

In this paper, an adjustable threshold algorithm for dynamic thermal management of multicore processors is presented. The proposed algorithm considers current workload and physical feature of cores. An appropriate temperature threshold is extremely important in DTM due to having great effect on system temperature and performance. Experimental results based on practical benchmarks (SPEC CPU2006) running on a desktop platform (Intel Core i7-2600) indicate that our algorithm can overcome Linux standard scheduler, TAS, and PDTM with negligible performance overhead. For the future work, we will test our scheme in different platforms with various benchmark to verify its scalability.

REFERENCES

[1] J. Kong, SW Chung, and K. Skadron, "Recent thermal management techniques for microprocessors," *ACM Computer Survey*, vol. 44, no. 3, pp. 13:1-13:42, 2012.

[2] I. Yeo, C. C. Liu, and E. J. Kim, "Predictive dynamic thermal management for multicore systems," *in DAC*, pp. 734-739, 2008.

[3] I. Yeo, E. Jung Kim, "Temperature-aware scheduler based on thermal behavior grouping in multicore systems," *in DATE*, pp. 946-951, 2009.

[4] Lm sensors linux hardware monitoring [Online]. Available: http://www.lm-sensors.org

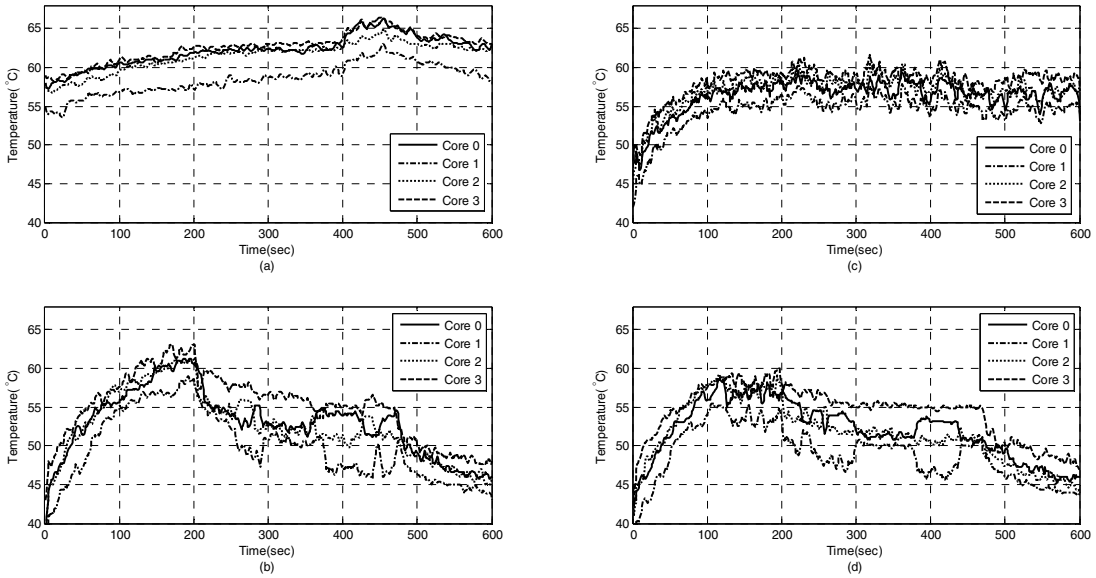

Figure 1 - Cores temperature for (a) Linux standard scheduler, (b) PDTM, (c) TAS, and (d) ATDTM