A novel CPU P-State driver for better thermal control with improved power/performance trade-off

by

Pavan Ravi Teja Uppu

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
December 12, 2015

Keywords: PID control, Thermald, Model Predictive Control (MPC), Thermal Headroom

Approved by

Sanjeev Baskiyar, Chair, Associate Professor of Computer Science and Software Engineering
Xiao Qin, Associate Professor of Computer Science and Software Engineering
Dean Hendrix, Associate Professor of Computer Science and Software Engineering

Abstract

Power Consumption and thermal emergencies in multi-core processors are now a major bottleneck due to increased heat dissipation caused by running intensive programs on densely integrated systems. High temperatures lead to unreliable and short lifespan of electronic devices. Currently, mechanisms are already in place so that whenever CPU reaches a cut-off temperature, the BIOS increases fan speed and conducts thermal throttling (adjusts the clock duty cycle and/or reduces the operating frequency/voltage). The problem of thermal control is exacerbated with the new Intel turbo boost, which opportunistically raises frequency leading to temperature spikes. Thus, there is increased need to control temperature at a set point via dynamic voltage and frequency scaling. Fortunately, modern CPUs provide P-States to operate it at various voltage-frequency pairs. P-States control temperature with the minimum loss in performance with the help of Proportional, Integral and Derivative (PID) based control. We implemented a Thermal head room based P-state driver (THBD) to reduce violations beyond target temperature. The THBD algorithm increases P-state only when there is enough thermal headroom, thus maintaining a steady temperature below the cut-off temperature, which could improve reduction in peak temperature and energy consumption by 1-4°C and 0.1-1.5 KJoules respectively with 0-7% increase in run-time penalty when evaluated with SPEC 2006 benchmark suite against PID based control.

Acknowledgments

I am deeply thankful to my advisor Dr. Sanjeev Baskiyar, for his careful guidance, persistence and perseverance through out this research. This thesis undoubtedly is impossible without his vision and help in academic writing. I would like to thank Dr. Xiao Qin and Dr. Dean Hendrix for being part of my thesis committee and for their guidance. I would like to thank Dr. Weikuan Yu, for his guidance and support during intial stages of my study. I would like to thank Dr. Hundley and Dr. Ku for awarding me a Teaching Assistant position.

On this occasion, I would like to remember and thank few more people in my life with out whom my masters at Auburn University would have been a distant dream. First of them all, I am deeply indebted to my cousin Mr. Naga Veeram for being more than a cousin and helping me in every cause. With all the respect from the bottom of my heart, I would like to thank my uncle and aunt, Sada Siva Rao and Chadrvathi. I would like to thank all my friends and well wishers, who just did not find place in this acknowledgement but in my heart.

Finally, I would like to thank my parents, Satyanarayana and Manimala, who endured suffering to provide good life, quality education for their children and they definitely deserve more than just thanks.

This thesis and further success if any, would undoubtedly be dedicated to my parents and my loving sister.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Power consumption and thermal emergencies in multi-core processors have become a major issue in personal computers as well as servers. The problem of task scheduling now includes thermal aware scheduling in order to conserve energy and increase lifetime of electronic circuits. A 10°C increase in overall temperature reduces the life span of electronic devices by half [1]. The cost for cooling and packaging also increases with increasing power [2,3]. Previously DTM, DVFS were used to control temperature once they reached the threshold [4] but with some loss of performance. Thermal aware strategies have emphasized distributing the performance both temporally and spatially to mitigate temperature [5]. Intel uses a thermal daemon based on a closed loop PID, controlling the P-States explained later in preliminaries section for reducing thermal emergencies. A brief description of Intel thermal daemon from [6] is explained in this paper and shown in Figure 1.1. For monitoring and controlling core temperatures Intel's Linux Thermal Daemon offers a viable solution. As the demand for high performance is growing, nowadays thinner and smaller computing systems such as laptops, ultra-books are being used for running HPC applications, which in turn demands innovative ways to monitor and reduce system heat dissipation. Based on temperature, the BIOS controls the system fan. It increases or decreases fan speed and can do thermal throttling accordingly. Thermal throttling adjusts the duty cycle of the processor clock or reduces the operating frequency, voltage. Such controls significantly impact performance [7]. When the maximum specified CPU temperature settings are exceeded performance loss occurs due to the intervention of BIOS, In order to prevent such an effect, the thermal daemon proactively cools the CPU by managing performance and thermal states. For controlling these states, P-state drivers and other cooling devices are being used in the Linux kernel developed

by Intel. The block diagram for such an implementation is shown in Figure 1.1 which has been reproduced from [7]. Although these P-states perform quite well for the power and performance trade-off, they are not suitable for temperature control at a set point because they are discrete in nature and makes temperatures oscillate below and above the set point. The official information as to how many states are available in a system are not yet published and we know that they are limited in number [8]. After trying several approaches and experiments, we determined that that a modified approach based on thermal headroom works well for temperature and offers energy savings with low schedule length penalty. In this research, a P-State driver was built which distributes performance temporally while governing the target temperature with minimal modifications to the underlying operating system. Although temperature could be predicted using a temperature prediction model as in [9–11] but we used the thermal virtual file system (i.e., /sys/class/hwmon) in Advanced Configuration and Power Interface (ACPI) to directly read the temperature. Temperature reaction time (rise and fall time) was found to be of a tenth of a second similar to that in [12]. Thermal virtual file system reports temperature every 2 seconds which we choose as the sampling interval. Hardware performance counters, thermal virtual file system were made use of in gathering data for temperature, performance and power. Using the above data, the correlation between the cpu maximum performance and temperature was exploited. Thermal Head Room analysis helps determine if the temperature may exceed the target temperature in future. The Thermal Headroom approach takes into account the discrete nature of the P-States. Transition to higher P-States is made only when the headroom is available. This approach reduces thermal oscillations and conserves energy by avoiding unnecessary cooling. We tested our approach on HPC applications such as SPEC CPU 2006 benchmarks. The overall peak temperature was reduced with minimum schedule length penalty. The key idea behind our approach was to limit performance when temperature crosses the set point. As power is proportional to the square of processor frequency, there is conservation of energy [13, 14].

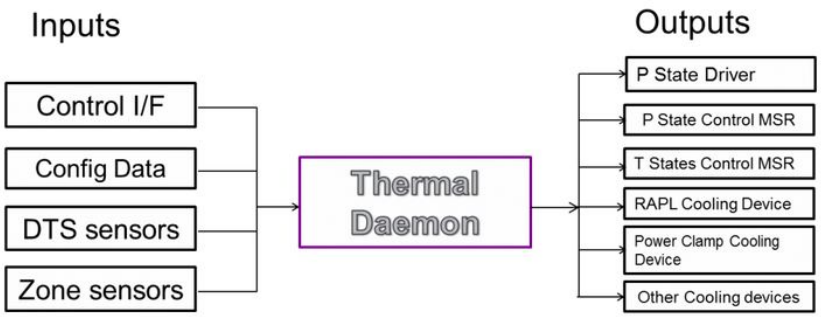


Figure 1.1: Thermald Block Diagram

The remainder of this thesis is organized as follows. Chapter 2 discusses related work in the area of temperature aware scheduling. Chapter 3 gives preliminaries for this research. In Chapter 4, we present our Thermal Headroom based Performance Driver. Chapter 5 presents our experimental analysis and results. Finally, in Chapter 6 we conclude this thesis.

Chapter 2

Related Work

Many researchers in thermal aware scheduling have explored the use of DVFS and adaptive process scheduling in reducing the core temperature. Choi, et al., [12] proposed different methods for temperature control like Heat Balancing, Deferred execution of hot tasks, cool loops, core hopping. Their potentials to leverage temporal heat slack was also discussed.

Lee, et al., [5] proposed a thermal aware scheduler collaborating with OS and architecture where an incoming thread is classified based on variation of the core temperature dynamically and scheduled at appropriate time on a core based on its temperature. This scheduling method succeeds only when there is a good mixture of cold and hot processes. When the core temperature reaches specified maximum, the cold jobs are always preferred jeopardizing execution of hot jobs.

Yue, et al., [15] proposed a thermal aware feedback control scheduling for Soft Real-Time Systems using cache-miss ratio as feedback where they could set temperature at a reference point, with a tight feedback loop controlling frequency via cache-miss ratio. However, in presence of noise due to fan and heat from other cores in case of a multi core general purpose computing CPU, the frequency-miss rate model does not work as intended. Thus, a direct control over frequency such as DVFS is indeed necessary.

Jayaseelan and Mitra, [16] proposed a Thermal Aware Scheduler for Embedded Processors. They developed a temperature-aware scheduling scheme which exploits the variability between soft real-time and best effort applications to maintain the system temperature below

a desired level while satisfying requirements such as throughput and fairness in temperature-constrained systems. The thermal model proposed as above would not support rapid temperature change because in a turbo state, the processor frequencies are opportunistically scaled [17] which in-turn leads to abrupt changes in temperature.

Li, et al., [18] proposed a fully loaded thermal-aware process scheduling on priority based scheduling for the hot process and cold process, which were classified based on instructions per cycle. Their techniques targeted scenarios common to high-performance computing where processors are fully loaded. However when the system runs out of cool process, there is a rapid temperature increment, then DVFS takes corrective action to maintain temperature below cut-off suggesting that process scheduling cannot be a complete alternative to DVFS when the temperature emergencies could happen at a granularity of millisecond.

Other important research in the thermal aware scheduling was to tackle the problem of performance optimization of set of tasks executing on a processor under thermal constraints. Zhang and Chatha, [11] addressed latency minimization of a set of periodic tasks with discrete voltage/frequency states under thermal constraint. They formulated it as a knapsack problem by incorporating process sleep time in the problem. The solution technique is to assign voltage states for each job and selecting a process sleep time using dynamic programming approach. A polynomial time solution within 1% quality bound of the optimal solution was proposed. Fundamental problems such as performance under a thermal constraint with discrete DVFS which were not focused before were addressed. But it did not take account of leakage power due to supply voltage and DVFS state transition overhead. As a consequence, overall energy consumption may not be minimized.

Rajan and Yu, [19] showed that often constant policies are better than Zig-Zag policies except in case where one is forced to Zig-Zag, where the system can operate in only few discrete states. The zig-zag scheme executes jobs at highest speed till cut-off temperature and then speed is reduced to allow the processor to cool. They have formulated an optimal Zig-Zag policy by selecting processor speeds for jobs with different priorities for reduction

in schedule length. One such problem associated with discrete P-states are temperature violations which occur while trying to optimize performance under at a particular target temperature.

In this paper we used the real time data to predict and prevent future thermal violations while not compromising on the performance of the applications.

Chapter 3

Preliminaries

3.1 States of a Processor and Intel Speed Step technology

Thermal and power management has been standardized in Enhanced Intel Speed Step Technology by centralizing hardware implementation using Package Control Unit (PCU) explained later in subsection 3.2. It provides better features for controlling the system transition between states shown in Figure 3.1. To manage increasing constraints on power and thermal budgets, performance state transitions can be dynamically controlled.

A brief description of Enhanced Intel Speed Step Technology from [20] is explained here. Multiple Voltage Identification (VID) pins, which signal desired voltage, are connected to voltage regulator in processors featuring Enhanced Intel SpeedStep Technology. These VID pins can be set to a demand voltage for the CPU. The voltage regulator subsequently provides this VID output to the core processor. Embedded operating systems and applications can easily change and control operating states using centralized implementation of real-time frequency/voltage transitions in the processor's module specific registers. A CPU can be put to different power states, depending on the current workload. These states are determined by the active parts of the CPU. C0 is active mode running instructions, C1, C1E are auto halt mode scaling frequency and voltage opportunistically. C3 corresponds to L1/L2 caches flush and clock off. G1 (Sleeping mode) encompasses S1 corresponding to standby mode, S3 Suspend to Ram (STR) and S4 Hibernate mode. G3 is when the system is mechanically switched off. Processor P-states are defined as frequency/voltage operating states. All P-states are sub-states of C0 and the processor actually executes instructions in all P-States unlike other states, such as C1, C2 and C3. The maximum performance and the most power

consuming state is P0. The whole system states and voltage, frequency varying with these states are illustrated in Figures 3.1 and 3.2 which are reproduced from [21].

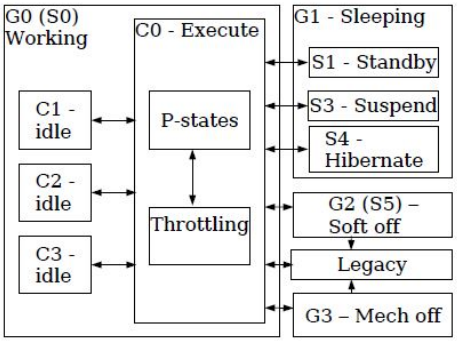


Figure 3.1: States of CPU

C-State	P-State	MHz	Volts	Watts
C0	P0	1600	1.484	24.5
	P1	1300	1.388	22
	P2	1100	1.180	12
	P3	600	0.956	6
C1, C2	from P0	0	1.484	7.3
	from P3	0	0.956	1.8
C3	from P0	0	1.484	5.1
	from P3	0	0.956	1.1
C4	(any)	0	0.748	0.55

Figure 3.2: C-State and P-State processor power

3.2 Package Control Unit (PCU)

Package Control Unit is an embedded controller in Intel's core i7 processor for running power management firmware and communicates with all cores monitoring and regulating conditions like Package Temperature and power states using CPU voltage and frequency. The decision of switching among power management states (P/T/C/S) in order to effectively balance power and performance is taken by the PCU firmware dynamically [8]. The Core i7s entire microprocessor package, and the overall computer system are in turn controlled through these states. The firmware for the power management software can be upgraded

and is scalable [8]. Those components which are not active like last level caches are switched off by the PCU [22].

Algorithm 1 Thermal Head Room Based P-State Driver

```
1: const P-state-max; P-state-min;                          ▷ max and min possible states
2: procedure THERMAL HEADROOM(int Target-temp)
3:     int T; lookup-headroom=0; THR=0;          ▷ Temperature; flag; Thermal headroom
4:     while (1) do
5:         Wait(δt)                                                  ▷ polling interval
6:         T ← Read Temperature
7:         if (T > Target-temp) and (P-state > P-state-min) then
8:             P-state−−                                              ▷ cool CPU
9:             lookup-headroom ← 1                  ▷ set flag to compute headroom later
10:        else if T < Target-temp then
11:            if lookup-headroom and (P-state < P-state-max) then
12:                P-state++                  ▷ switch to upper state for computing THR
13:                Wait(δt)
14:                T′ ← Read Temperature
15:                P-state−−                  ▷ switch to lower state after computing THR
16:                THR ← T′ − T                ▷ Compute headroom and turn-off flag
17:                lookup-headroom ← 0
18:            else if (Target-temp - T) ≥ THR and (P-state < P-state-max) then
19:                P-state++                          ▷ headroom available, so speed-up
20:            end if
21:        end if
22:    end while
23: end procedure
```

Chapter 4

Thermal Headroom based P-State Driver

This research controls the CPU temperature and it's variations by changing P-States using an approach that is inspired by Model Predictive Control (MPC). The description of MPC in [23] has been adopted and re-explained for Thermal Headroom approach. MPC is a linear algebra method for predicting the results of a sequence of control variable changes. After a few initial observations, the controller can use a control sequence which produces the desired output. This method could be compared to "look ahead" in chess. Specific knowledge

of the process could be used to optimize the best long-term output by forecasting the results of action in future. MPC methods could prevent actions taken by conventional PID control to achieve short term goals, which are costly in the long-term. Deviations from the desired output y_d, either specified by another mathematical model or reference trajectory, produce an error function $e(t) = y(t)$ - $y_d(t)$ for increments of control actions $\Delta u(t) = u(t)$-$u(t - \Delta t)$. Here $y(t)$ is the output and $u(t)$ is the control action. In case of P-State control $p + 1^{th}$ state and p^{th} state can be considered as control actions and y_d(t) is the desired target-temperature and $y(t)$ is the current temperature. Thermal Head Room based algorithm is inspired by MPC control. However, it uses reference trajectory for measurements of $y(t)$, obtaining a reference trajectory through a system model may not be accurate in case of processors where the temperature varies both with program transformation and input control signal. Thus, We measure $\Delta y(t)$ as headroom for $\Delta u(t)$. We switch to higher P-State only when $e(t)$ is less than or equal to this headroom. We switch to a lower P-State whenever the observed temperature is greater than target temperature. Such an approach prevents unnecessary oscillations of temperature around the target temperature because the P-State is increased only if it is predicted that doing so will not increase the temperature above the target temperature.

This approach is necessary as the P-States are discrete. The Thermal Head Room based P-State driver provides a history based learning where thermal head room is measured during the time when the P-State is boosted. The simulated example in figure 4.1 shows the functioning of the algorithm. When the temperature overshoots the target temperature, the Thermal Headroom is measured to be 5°C and 4°C in two instances as shown in figure 4.1, which is the temperature change occurred due to change in P-State. The algorithm learns that there is an overshoot of temperature with an increase in P-State, thus an increase in P-State is performed only when this thermal headroom is available next time.

The Thermal Head Room Based P-State Driver algorithm waits for polling interval on line 6 and the temperature is read on line 7, if the observed temperature is greater than target

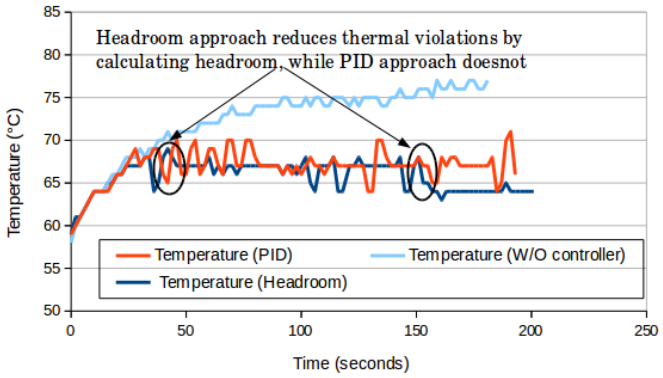


Figure 4.1: Temperature vs Run-time comparison of Headroom, PID and without control

temperature, the P-State is simply reduced to next lower State as in line 9 and a lookup-headroom bit is set and the temperature if found to be less than target temperature on line 11 based on lookup-headroom the P-State is raised and Thermal Headroom is calculated on line 16.In consecutive iterations, the difference between target temperature and current temperature if greater than measured headroom will lead to increment in P-State on line 19.

Chapter 5

Experimental Results

5.1 Experimental Setup

The PID and Thermal Headroom based techniques were coded in C and the experiments were performed on Intel i7-2630 quad core processor of HP Probook 4530 Laptop running Ubuntu 14.04 LTS at base frequency of 2 Ghz with 4 GB RAM. L1 cache comprises of 4 x 32 KB 8-way set associative instruction caches and 4 x 32 KB 8-way set associative data caches, L2 cache is 4 x 256 KB 8-way set associative and L3 cache is a 6 MB 12-way set associative shared. We obtained the temperature through system files /sys/class/hwmon/hwmon1/temp2_input and energy was measured using Intel RAPL and perf tool which reports energy consumed by the package with in specified poll interval. We choose 10 P-States and poll-interval of 4 seconds similar to the default settings of thermald software. The system performance percentage from 100 to 0 was split into 10 ranges each corresponding to a P-State, which inturn affects frequency and voltage where the frequency would be varied from 3 Ghz to 300 Mhz. The values of voltage and frequency pairs associated with P-States of core i7 are not yet published by Intel [8]. We used task set in Linux to pin the processes to a single core, making it fully loaded while the OS runs on the remaining cores. Ambient room temperature was set to 21 °C. We set the target temperatures at 67°C, 70°C and 74°C which correspond to day to day use of the laptop, and ran 26 SPEC CPU2006 benchmarks. At the target temperature of 67°C individual benchmarks were enough to create sufficient enough thermal stress. At 70°C and 74°C we grouped the benchmarks as shown in table 5.1, so that the outcome at target temerature could be more prominent with longer run-times. The focus was to stress the algorithm as much as possible by overloading multiple processes on a core using the taskset command. We have included run-time graphs of the benchmarks where

in the difference between PID based and the Thermal Headroom based drivers is clearly shown. Energy conservation was significantly better when multiple processes were run with not much schedule length penalty at 70°C and 74°C.

Table 5.1: Benchmarks setup for different Target temperatures

Target temperature	Benchmarks	Group
67°C	lbm	Group 1
	sphinx	Group 2
	povray	Group 3
	namd	Group 4
70°C	calculix+omnetpp	Group 5
	perlbench+soplex	Group 6
	h264ref+astar	Group 7
	gamess+gromacs	Group 8
	tonto+leslie3d	Group 9
74°C	bzip2+gemsfdtd	Group 10
	bwaves+gcc	Group 11
	sjeng+cactusadm	Group 12
	lbmquantum+wrf	Group 13
	hammer+gobmk	Group 14
	xalanbmk+zuesmp	Group 15

5.2 Results for target temperature 67°C

Using Thermald, the PID based approach was evaluated under different scenarios under stress using different SPEC CPU 2006 benchmarks. We see from Figures 5.2, 5.3, 5.4, 5.5 that there were temperature swings until the temperature gets stabilized at the target temperature of 67°C. An XML configuration file has been provided in Thermald, where one can tune the PID controller with different gains but cooling due to fan, the heat from other cores, program transformations can drive the control variable to shift from equilibrium thus leading to instability. In such an unstable state, The THBD approach auto-tunes itself using past behavior of the temperature and making an appropriate decision to switch P-States. It waits every time the temperature goes below the target temperature until the thermal headroom is available before the P-State is increased. From the run-time graphs in 5.2, 5.3,

5.4, 5.5 we see that temperature violations in the headroom approach are fewer compared to the PID based approach. Furthermore, Each P-State transition takes time; by making fewer transitions to higher P-States, the headroom approach incurs small run time penalty when compared to PID based approach. This mechanism provides system reliability by maintaining temperature below target temperature even in turbo mode where processor is overclocked in the highest performance state.

One more advantage with this mechanism is the reduction of run-time power consumption. As the temperature increases leakage current in the insulator dielectric in the CPU increases [24] as modeled by the Poole-Frenkel effect. This effect has been illustrated in figure 5.1 which has been reproduced from [24]. Reducing the temperature spikes lead to energy conservation amounting approximately up to 10 Joules/spike as seen in the run-time graphs in 5.2, 5.3, 5.4, 5.5. The performance loss for Thermal headroom approach is not significant during these phases as seen from Figures 5.2, 5.3, 5.4, 5.5 and is compensated by the P-State transition time. A PID control is generally used for all the active and cooling devices such as P-States, Fan. However P-State transitions further optimize energy by limiting the temperature just below the set-point. The energy savings obtained from leakage current by maintaining temperature below set-point combined with dynamic energy savings were plotted and are shown in Figures 5.2, 5.3, 5.4, 5.5. Figures 5.6, 5.7, 5.8 show run-time, energy and peak temperature for PID, Baseline(W/O Controller) and Thermal headroom based approach. The increase in run-time penalty when lbm was run was observed to be 1.15% with a 3°C reduction in temperature and 1.5% reduction in energy consumption. When sphinx was run the increase in run-time penalty was 4.3% with reduction in peak temperature and energy of 3°C and 1% respectively. When povray was run the increase in run-time penalty was 4% with reduction in peak temperature and energy of 2°C and 3% respectively. When namd was run, the increase in run-time penalty was 0.4% with reduction in peak temperature and energy of 2°C and 1.3% respectively.

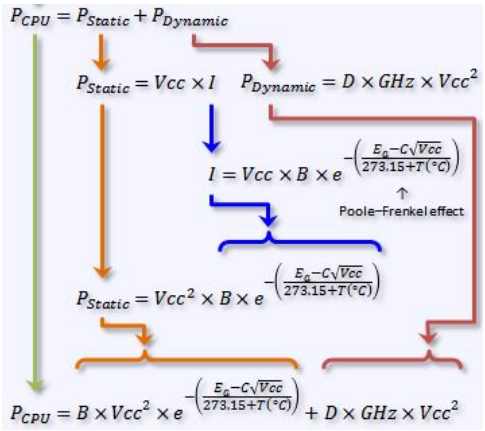


Figure 5.1: Power consumption due to Poole-Frenkel effect

5.3 Results for target temperature 70°C

The increase in schedule length penalty when calculix, omnetpp were scheduled together was observed to be 3.9% while contributing to 3°C reduction in temperature and 2.5% reduction in energy consumption. When perlbench, soplex were scheduled together the increase in schedule length penalty was 6.4% with reduction in peak temperature and energy of 1°C and 7.2% respectively. When h264ref, astar were scheduled together the increase in schedule length penalty was 5.3% with reduction in peak temperature and energy of 3°C and 5.9% respectively. When gamess, gromacs were scheduled together the increase in schedule length penalty was 4.46% with reduction in peak temperature and energy of 2°C and 4.88% respectively. When tonto, leslie3d were scheduled together the increase in schedule length penalty was 1% with reduction in peak temperature and energy of 3°C and 4.7% respectively. One striking observation is that when benchmarks are scheduled in groups together without much change in schedule length penalty, the energy conservation is improved with Head Room based P-state Driver without any significant schedule length penalty.

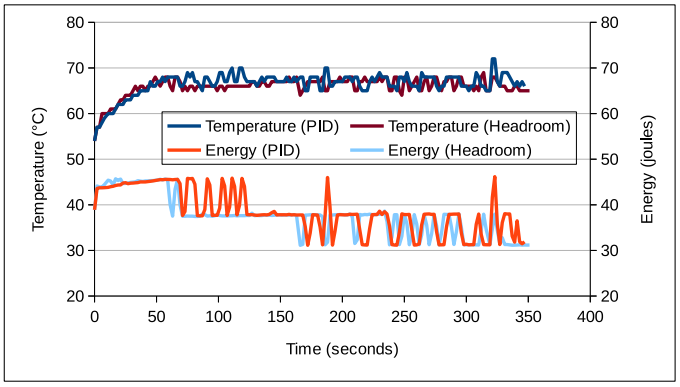


Figure 5.2: CPU Temperature and Energy vs Time for LBM

5.4 Results for target temperature 74°C

The increase in schedule length penalty when bzip2, gemsfdtd were scheduled together was observed to be 0.2% while contributing to 1°C reduction in temperature and 8% reduction in energy consumption. When bwaves, gcc were scheduled together the increase in schedule length penalty was 1.4% with reduction in peak temperature and energy of 2°C and 0.3% respectively. When sjeng, cactusadm were scheduled together the increase in schedule length penalty was 1.3% with reduction in peak temperature and energy of 2°C and 3.8% respectively. When lbmquantum, wrf were scheduled together the increase in schedule length penalty was 3.4% with reduction in peak temperature and energy of 1°C and 4% respectively. When hammer, gobmk were scheduled together the increase in schedule length penalty was 2.3% with reduction in peak temperature and energy of 2°C and 3% respectively. When xalanbmk, zuesmp were scheduled together the increase in schedule length penalty was 1.2% with reduction in peak temperature and energy of 1°C and 2.2% respectively.

Table 5.2: Results at target temperature 67°C

Group	Run-time (seconds)			Peak temperature (°C)			Energy consumed (KJ)		
	PID	Headroom	Base	PID	Headroom	Base	PID	Headroom	Base
G1	347	351	325	72	69	80	6.66	6.56	7.58
G2	717	748	657	73	70	83	11.75	11.62	14.34
G3	193	201	181	71	69	77	3.55	3.44	3.84
G4	486	488	454	71	69	78	8.22	8.11	9.28

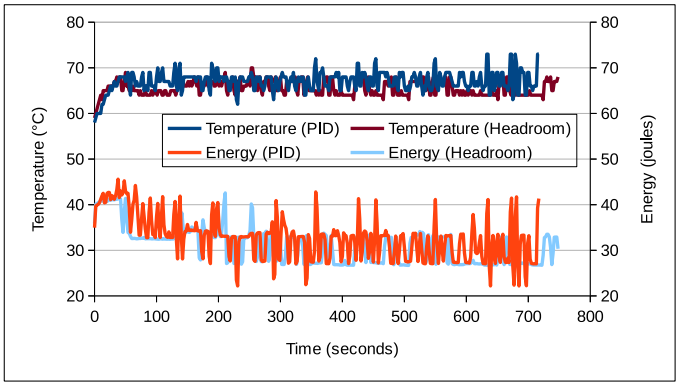


Figure 5.3: CPU Temperature and Energy vs Time for SPHINX

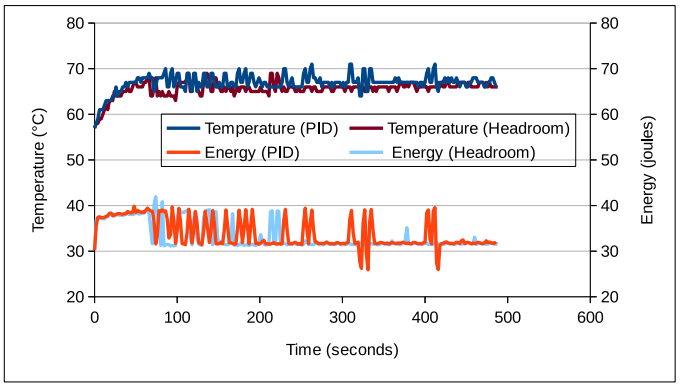


Figure 5.4: CPU Temperature and Energy vs Time for NAMD

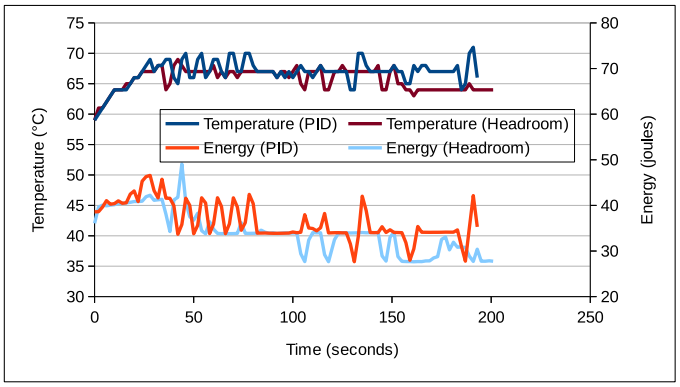
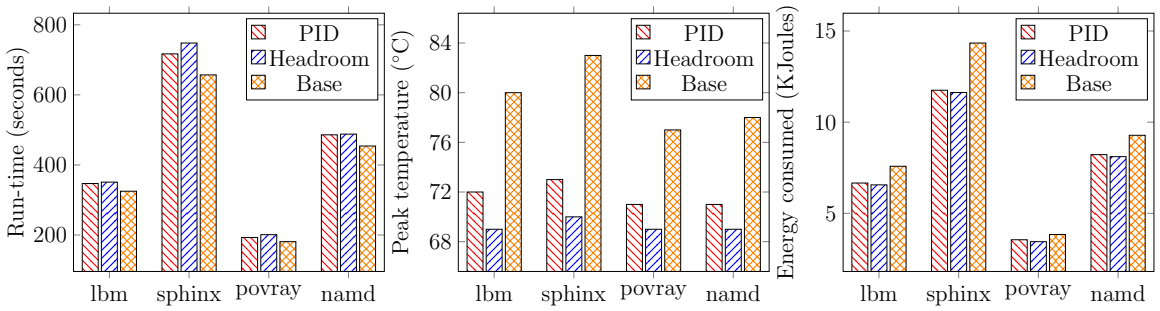
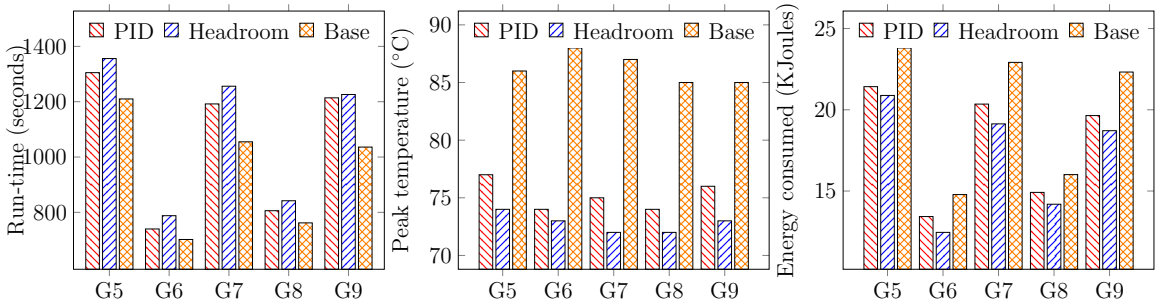


Figure 5.5: CPU Temperature and Energy vs Time for POVRAY



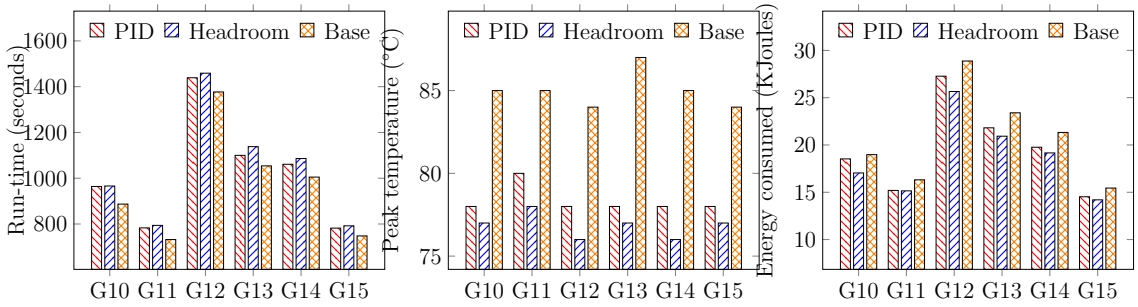
(a) Run-time (b) Peak temperature (c) Energy consumption

Figure 5.6: Headroom vs Base vs PID based P-state driver at 67°C target temperature



(a) Run-time (b) Peak temperature (c) Energy consumption

Figure 5.7: Headroom vs Base vs PID based P-state driver at 70°C target temperature



(a) Run-time (b) Peak temperature (c) Energy consumption

Figure 5.8: Headroom vs Base vs PID based P-state driver at 74°C target temperature

Table 5.3: Results at target temperature 70°C

Group	Run-time (seconds)			Peak temperature (°C)			Energy consumed (KJ)		
	PID	Headroom	Base	PID	Headroom	Base	PID	Headroom	Base
G5	1305	1356	1210	77	74	86	21.42	20.88	23.81
G6	740	788	702	74	73	88	13.43	12.45	14.78
G7	1192	1256	1055	75	72	87	20.35	19.13	22.92
G8	806	842	762	74	72	85	14.92	14.19	16.01
G9	1214	1226	1036	76	73	85	19.64	18.71	22.32

Table 5.4: Results at target temperature 74°C

Group	Run-time (seconds)			Peak temperature (°C)			Energy consumed (KJ)		
	PID	Headroom	Base	PID	Headroom	Base	PID	Headroom	Base
G10	964	966	887	78	77	85	18.53	17.04	18.98
G11	783	794	732	80	78	85	15.20	15.15	16.31
G12	1439	1459	1377	78	76	84	27.27	25.65	28.88
G13	1100	1138	1054	78	77	87	21.81	20.93	23.40
G14	1061	1086	1005	78	76	85	19.76	19.16	21.325
G15	782	792	748	78	77	84	14.52	14.20	15.44

Chapter 6

Conclusion and Future work

In this thesis, we investigated the trade-off among schedule length, temperature and energy of a new Thermal Head Room based P-State driver. We could significantly reduce temperature overshoots beyond set-point and the energy conservation is observed as a side affect of this approach. To the best of our knowledge, this research is first to analyze Thermald for temperature violations beyond reference temperature and propose a new mechanism for reducing violations under a thermal constraint. We have conducted an extensive architectural study of the PID and Head room based thermal drivers. In future we would like to extend the approach to distributed processing with a high performance computing perspective.

Bibliography

[1] L. Yeh and R. Chu, "Thermal management of microelectronic equipment: Heat transfer theory," *Analysis Methods, and Design Practices, ASME, New York*, 2002.

[2] S. Gunther, F. Binns, D. M. Carmean, and J. C. Hall, "Managing the impact of increasing microprocessor power consumption," *Intel Technology Journal*, vol. 5, no. 1, pp. 1–9, 2001.

[3] H. F. Hamann, A. Weger, J. A. Lacey, Z. Hu, P. Bose, E. Cohen, and J. Wakil, "Hotspot-limited microprocessors: Direct temperature and power distribution measurements," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 1, pp. 56–65, 2007.

[4] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 171–182. [Online]. Available: http://dl.acm.org/citation.cfm?id=580550.876439

[5] C.-Y. Lee, S.-J. Yang, and R.-G. Chang, "Thermal-aware scheduling collaborating with os and architecture," in *42nd International Conference on Parallel Processing*. IEEE, 2013, pp. 1044–1051.

[6] L. Michael, "Intel releases linux thermal daemon," 2013. [Online]. Available: http://www.phoronix.com/scan.php?page=news_item&px=MTM2OTk

[7] S. Pandruvada, "Linux thermal daemon," Intel. [Online]. Available: https://01.org/linux-thermal-daemon/documentation/introduction-thermal-daemon

[8] "Intel and core i7 (nehalem) dynamic power management." Arizona State University, 2011. [Online]. Available: https://impact.asu.edu/cse591sp11/Nahelempm.pdf

[9] Y. Fu, N. Kottenstette, Y. Chen, C. Lu, X. D. Koutsoukos, and H. Wang, "Feedback thermal control for real-time systems," in *16th IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2010, pp. 111–120.

[10] I. Yeo and E. J. Kim, "Temperature-aware scheduler based on thermal behavior grouping in multicore systems," in *Proceedings of the conference on Design, Automation and Test in Europe*. European Design and Automation Association, 2009, pp. 946–951.

[11] S. Zhang and K. S. Chatha, "Approximation algorithm for the temperature-aware scheduling problem," in *10th IEEE/ACM Proceedings of the international conference on Computer-aided design*. IEEE Press, 2007, pp. 281–288.

[12] J. Choi, C.-Y. Cher, H. Franke, H. Hamann, A. Weger, and P. Bose, "Thermal-aware task scheduling at the system software level," in *Proceedings of the 2007 International Symposium on Low Power Electronics and Design*. New York, NY: ACM, 2007, pp. 213–218. [Online]. Available: http://doi.acm.org/10.1145/1283780.1283826

[13] S. Baskiyar and R. Abdel-Kader, "Energy aware dag scheduling on heterogeneous systems," *Cluster Computing*, vol. 13, no. 4, pp. 373–383, 2010.

[14] S. Baskiyar and K. K. Palli, "Low power scheduling of dags to minimize finish times," in *High Performance Computing-HiPC 2006*. Springer, 2006, pp. 353–362.

[15] J. Yue, T. Zhang, Y. Liu, B. Quan, and C. Tianzhou, "Thermal-aware feedback control scheduling for soft real-time systems," in *IEEE 9th International Conference on Embedded Software and Systems, IEEE 14th International Conference on High Performance Computing and Communication, Liverpool, United Kingdom*, June 2012, pp. 1479–1486.

[16] R. Jayaseelan and T. Mitra, "Temperature aware scheduling for embedded processors," in *22nd International Conference on VLSI Design, New Delhi, India*, Jan 2009, pp. 541–546.

[17] O. Lempel, "2nd generation intel core processor family: Intel core i7, i5 and i3," 2011. [Online]. Available: http://download.intel.com/newsroom/kits/embedded/pdfs/2nd_Gen_Intel_Core_i3_i5_i7_PlatformBrief.pdf

[18] D. Li, H.-C. Chang, H. K. Pyla, and K. W. Cameron, "System-level, thermal-aware, fully-loaded process scheduling," in *IEEE International Symposium on Parallel and Distributed Processing*, 2008, pp. 1–7.

[19] D. Rajan and P. S. Yu, "On temperature-aware scheduling for single-processor systems," in *Proceedings of the 14th International Conference on High Performance Computing (HiPC), Goa,India*. Springer-Verlag, 2007, pp. 342–355. [Online]. Available: http://dl.acm.org/citation.cfm?id=1782174.1782214

[20] "Enhanced intel speedstep technology for the intel pentium-m processor," White Paper, Intel, 2004. [Online]. Available: http://download.intel.com/design/network/papers/30117401.pdf

[21] L. Brown, "Acpi in linux," in *Linux Symposium*, vol. 51, 2005.

[22] A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann, "Power management architecture of the 2nd generation intel core microarchitecture, formerly codenamed sandy bridge," p. 0, 2011. [Online]. Available: http://www.hotchips.org/wp-content/uploads/hc_archives/hc23/HC23.19.9-Desktop-CPUs/HC23.19.921.SandyBridge_Power_10-Rotem-Intel.pdf

[23] "Model predictive control." [Online]. Available: https://controls.engin.umich.edu/wiki/index.php/MPC

[24] "Effect of temperature on power-consumption with the i7-2600k." [Online]. Available: http://forums.anandtech.com/showthread.php?t=2200205