

Thermal-Aware Scheduling for Peak Temperature Reduction with Stochastic Workloads

SHAOBO LIU¹ AND MEIKANG QIU²

¹Department of ECE, State University of New York at Binghamton, Binghamton, NY 13902, sliu5@binghamton.edu

²Department of ECE, University of Kentucky, Lexington, KY 40506, mqiu@engr.uky.edu

Abstract

In this paper we proposed a thermal-aware scheduling algorithm for peak temperature reduction with stochastic workloads. The proposed approach consists of two steps. The first step constructs a job sequence based on the mean of the execution time of each job; and the constructed job sequence interleaves the execution of hot jobs and cool jobs. The hot job pushes up the chip temperature while the cool one cools down the chip. Accordingly the chip reduces the peak temperature. The second step exploits the job slack to cancel out the adverse thermal impact due to the uncertainty of the job execution time. The slacks are allocated among the hot jobs. Hot jobs whose execution time deviates more above the average execution time will get more slack, vice versa. Experimental results show that the proposed approach effectively avoids the thermal emergency for the applications with stochastic workloads by achieving the peak temperature reduction of 4.7°C on average.

1. INTRODUCTION

On-chip temperature is traditionally managed by advance packaging and cooling solutions. As the power density goes up exponentially, such solutions for temperature management is becoming prohibitively expensive [11]. In the past years researchers have been actively exploring alternative techniques to control on-chip temperature. The majority of those techniques are reactive. When the on-chip temperature is over some predefined threshold, a proper technique is applied to reduce the temperature or to keep the temperature below the critical temperature of the chip. Such techniques include dynamic power management (DPM), dynamic voltage and frequency selection (DVFS), dynamic thermal management (DTM), fetching throttling and task migrations, etc [9]. Those techniques are designed specifically for managing the thermal behavior of general purpose processors.

Leakage power is not negligible for 45nm process node and beyond [3, 4]; the positive feedback between leakage and temperature deteriorates the thermal behavior of a chip [9]. Embedded systems subject to the size constraints and have the limited air-cooling capability; as a consequence, the thermal challenges in embedded systems are even worse. Several thermal management techniques specifically for embedded systems have been discussed recently [1, 2, 5, 7, 8] for periodic deterministic application. However, the execution time of the application shows the stochastic properties in some cases [6] in the context of embedded systems.

In this paper, we are going to address the problem of minimizing the peak temperature of a job sequence with uncertain execution time from the statistical sense. Zhang et.al proposed a solution in [7] to the same problem, where DVFS is the only mechanism to manage the temperature. In this paper, we proposed an algorithm which combine the job sequencing and the voltage selection to achieve the peak temperature reduction. The proposed algorithm consists of two steps. A job sequence is constructed in

the first step to better off the thermal condition of the chip; the execution of the task is stretched and DVFS mechanism is applied in the second step to further improve the thermal behavior of the chip. The proposed algorithm is simple and it is easy to implement. Its computation complexity is also low.

The rest of the paper is organized as follows. The related work is presented in Section 2. The background is introduced in Section 3. The job sequencing algorithm and slack distribution algorithm are explained in section 4 and 5. The experimental results are presented and discussed in Section 6. And Section 7 gives summaries of the work.

2. BACKGROUND

In this section, we are going to introduce the DVFS-enabled processor model, the job model, and thermal model. All these models lay the foundation of our proposed algorithm. We start from the DVFS-enabled processor model.

2.1 Application Model

We consider a DVFS-enabled processor with M active states $\mathcal{M} = \{s_i | i = 1, 2, \dots, M\}$. In each state s_i , the processor has the correspondent operating voltage v_i and frequency f_i , respectively. We assume $f_{i+1} \geq f_i$; so the maximum operating frequency is f_M and the minimum one is f_1 . Notation $\rho_{i,j}$ denotes the power consumption of processor which executes job J_j at active state s_i .

The DVFS-enabled processor runs a sequence of independent periodic jobs $\mathcal{J} = \{J_j | j = 1, 2, \dots, N\}$. In this paper, we study applications with stochastic workload. The execution time of job J_j at active state s_M is denoted as W_j , which is a random variable; and the execution time of J_j at active state s_i can be calculated as $\frac{W_j f_M}{f_i}$, where $i = 1, 2, \dots, M - 1$.

For the sake of convenience, we assume W_j is a discrete random variable. The distribution of W_j is given by

$$P_{W_j}(W_j = w_{j,l}) = p_{j,l} \quad (1)$$

The average execution time $E(W_j)$ of J_j is $\sum p_{j,l} w_{j,l}$. Assume the arrival time and deadline of J_j are denoted as a_j and d_j , we define the slack time for J_j as

$$slack_j = d_j - a_j - E(W_j) \quad (2)$$

2.2 Thermal Model

We use a lumped RC thermal model to describe the thermal behavior of the processor [8] due to the duality between the one-dimensional heat transfer and the electrical RC circuit. The thermal capacitor and the thermal resistance of the processor are denoted as C and R , respectively. The initial temperature of processor is denoted as T_{init} . The temperature after the processor runs at power ρ for time t can be calculated as

$$T(t) = \rho R + T_{amb} - (\rho R + T_{amb} - T_{init}) e^{-\frac{t}{RC}} \quad (3)$$

where T_{amb} is the ambient temperature.

If the processor just keeps running the infinity copies of same job J_j , then we have

$$T_j(\sum W_j) = \rho_{M,j}R + T_{amb} - (\rho_{M,j}R + T_{amb} - T_{init})e^{-\frac{\sum W_j}{RC}} \quad (4)$$

Since $\sum W_j \rightarrow \infty$, and the above equation can be reduced to

$$T_{j,s} = \rho_{M,j}R + T_{amb} \quad (5)$$

where $T_{j,s}$ is called the steady temperature of job J_j , and $T_{j,s} = T_j(\infty)$. Note that the steady temperature of job J_j is referred to as the processor's temperature after an infinite sequence of job J_j is executed at the maximal operating frequency f_M .

The temperature change $\Delta T_j(W_j)$ of the processor after job J_j is executed can be calculated as

$$\Delta T_j(W_j) = T_j(W_j) - T_{init} = (T_{j,s} - T_{init})(1 - e^{-\frac{W_j}{RC}}) \quad (6)$$

3. JOB SEQUENCING WITH UNCERTAIN EXECUTION TIME

Running jobs in the proper sequence could effectively reduce the peak temperature with deterministic workloads [8]. In this section, we will introduce an algorithm applied to the stochastic workload. Equation(6) tells us even if the workload is stochastic, we can still interleave hot jobs and cool jobs so as to reduce the peak temperature of the job sequence. We first define the peak temperature of a job sequence; and then introduce the criteria to categorize jobs into either hot jobs or cool jobs.

3.1 Peak Temperature of Job Sequence

Assume the processor executes a job sequence $\mathcal{L} = \langle J_1, J_2, \dots, J_N \rangle$, the processor temperature after each job is executed can be calculated from Equation(5),

$$\begin{aligned} T_1(W_1) &= T_{1,s} - (T_{1,s} - T_{init_1})e^{-\frac{W_1}{RC}} \\ &\vdots \\ T_N(W_N) &= T_{N,s} - (T_{N,s} - T_{init_N})e^{-\frac{W_N}{RC}} \end{aligned} \quad (7)$$

Note that in Equation(7) the initial temperature of executing job J_{j+1} is the same as the final temperature after executing job J_j , i.e., $T_{init_{j+1}} = T_j(W_j)$, where $j = 1, 2, \dots, N-1$. The peak temperature of job sequence \mathcal{L} is defined as

$$peak(\mathcal{L}) = \max(T_1(W_1), T_2(W_2), \dots, T_N(W_N)) \quad (8)$$

$T_1(W_1), T_2(W_2), \dots, T_N(W_N)$ are random variables, so $peak(\mathcal{L})$ is also a random variable. From that definition, we know the peak temperature of a job sequence is also the peak temperature of the processor after a sequence of jobs are executed.

3.2 Problem Formulation

The input to our thermal-aware scheduling algorithm is a job set $\mathcal{J} = \{J_j | j = 1, 2, \dots, N\}$ which contains N jobs. The execution time, and average power consumption of job J_j are denoted as $W_j, \rho_{M,j}$ respectively, where $1 \leq j \leq N$. In this paper, our scheduling algorithm needs to construct a job sequence \mathcal{L} such that the peak temperature $peak(\mathcal{L})$ is minimized if the jobs are executed based on the sequence \mathcal{L} .

When workloads are deterministic, clearly there exist $N!$ possible sequences for N jobs. In the case that workloads are stochastic, there are numerous instances for each sequence and each instance may have different peak temperature; and an exhaustive search approach is unpractical. Therefore, new scheduling algorithm should be developed for stochastic workloads such that the peak temperature could be minimized in most cases. Before we explain our novel thermal-aware scheduling algorithm in details, some prerequisite should be introduced first.

3.3 Job Classification

From Equation(5), we know the steady temperature $T_{j,s}$ of job J_j is solely decided by the power consumption $\rho_{M,j}$ of the processor when job J_j is executed. On the other hand, the relative relationship between $T_{j,s}$ and T_{init} determines the temperature change direction of the processor after job J_j is executed, as shown in Equation(6). Before classifying jobs, we introduce two parameters: 1) the critical temperature $T_{critical}$ of the processor; the reliability and the performance of the processor will degrade if the processor operate over $T_{critical}$. 2) the preferred operating temperature T_{pref} of the processor, which is usually several degrees lower than $T_{critical}$ and the processor operates reliably around T_{pref} . In order to capture the power characteristic of each job, we assume that the processor has the same temperature T_{pref} before running each job. Jobs are classified into cool jobs or hot jobs based on the relative relationship between T_{pref} and the final temperature of the processor after the job is executed at the frequency f_M . We introduce two definitions to define what is a hot job and what is a cool job.

Definition 1 A job is called the **hot job** if processor's temperature goes up after the job is executed at f_M , given the initial temperature of the processor is T_{pref} .

Definition 2 A job is called the **cool job** if processor's temperature goes down after the job is executed at f_M , given the initial temperature is T_{pref} .

The key to the above stated two definitions is T_{pref} , which is the pivot of recognizing a job as hot job or cool job. The value of parameter T_{pref} should be set based on the system configuration. Note that the execution time of the job has no impact on whether the job should be put into cool job category or hot job category. The cool/hot job categorization reflects the characteristics of job's power profile. However, the execution time of the job is one of the important factors to regulate the magnitude of the temperature change, as shown in Equation(6). Note that a job is called the *mild job* if the temperature of processor keeps unchanged after it is executed at f_M with the processor's initial temperature set to T_{pref} . Since mild jobs do not change the processor's temperature, we focus on hot/cool jobs only in the following discussion.

In order to effectively reduce the peak temperature of the job sequence, we should first understand how each job changes the temperature of the processor after the job is executed.

3.4 Upper Bound of Expected Temperature Change

When workloads are stochastic, the execution time of each job is modeled as a random variable. Accordingly, the temperature change after the job is executed is also a random variable, as shown in Equation(6). The expectation of the temperature change can be calculated as

$$E(\Delta T_j(W_j)) = (T_{j,s} - T_{pref})(1 - E(e^{-\frac{W_j}{RC}})) \quad (9)$$

Function e^{-x} is a concave function; and according to Jensen's Inequality [12], we have,

$$|E(\Delta T_j(W_j))| \leq \Delta T_{j,ub} \quad (10)$$

where $\Delta T_{j,ub}$ is the upper bound of the temperature change magnitude and is defined as

$$\Delta T_{j,ub} = |T_{j,s} - T_{pref}|(1 - e^{-E(W_j)/RC}) \quad (11)$$

3.5 Job Sequencing with Stochastic Workload

In this subsection, we are going to introduce the first step of the thermal-aware scheduling algorithm which constructs a job sequence to improve the peak temperature. Equation(7) shows that

the hot jobs heat up the processor while the cool jobs cool down the processor if the initial temperature of the processor is T_{pref} . The processor temperature goes down if a cool job is scheduled after a hot job, which effectively reduces the peak temperature, comparing to the scheduling of putting two hot jobs together. Note that the processor temperature is higher than T_{pref} after the hot job is executed; then the temperature reduction is more than the value provided in Equation(6) after the subsequent cool job is executed. In order to avoid the temperature elevation, the scheduling algorithm should put the jobs with opposite thermal characteristics close to each other [8].

Workloads are stochastic, we introduce a metric α_j to represent the thermal characteristics of job J_j , where α_j is defined as

$$\alpha_j = \begin{cases} \Delta T_{j,ub}, & \text{for } J_j \text{ is a hot job} \\ -\Delta T_{j,ub}, & \text{for } J_j \text{ is a cool job} \end{cases} \quad (12)$$

α_j describes the thermal characteristic of job J_j . Statistically speaking, the large α_j states that job J_j has large power consumption; similarly the small α_j indicates the correspondent small power consumption.

In order to calculate α_j , we need to know $E(W_j)$, which can be easily obtained by

$$\mu_j = E(W_j) = \sum_{l=1}^{q_j} w_{j,l} p_{j,l} \quad (13)$$

The first step of the proposed thermal-aware scheduling algorithm is presented in Algorithm 1. The algorithm proceeds in a bottom-up manner by putting the jobs with opposite thermal characteristics together until a single job sequence is obtained. The jobs are first grouped into $\frac{N}{2}$ subsequences and each subsequence contains two jobs; then $\frac{N}{2}$ subsequences are similarly grouped into $\frac{N}{4}$ subsequences and so on. The algorithm terminates until a single sequence is obtained. Note that if N is an odd number, $\mathcal{L}_{\frac{N}{2}}$ in line 10 in Algorithm 1 is defined as $\mathcal{L}_{\frac{N}{2}} = \mathcal{L}_{\lfloor \frac{N+1}{2} \rfloor}$.

Algorithm 1 Job Sequence Construction Algorithm

Require: Job set $\mathcal{J} = \{J_j | 1 \leq j \leq N\}$

```

1: for  $j = 1, \dots, N$  do
2:    $\mathcal{L}_j = J_j$ 
3: end for
4: while  $N > 1$  do
5:   for  $j = 1, \dots, N$  do
6:     calculate  $\alpha_j$ 
7:   end for
8:   sort  $(\mathcal{L}_1, \dots, \mathcal{L}_N)$  based on  $\alpha_j$ 
9:   for  $j = 1, \dots, \frac{N}{2}$  do
10:     $\mathcal{L}_j = \mathcal{L}_j \bullet \mathcal{L}_{N-(j-1)}$ 
11:   end for
12:    $N = \lfloor \frac{N+1}{2} \rfloor$ 
13: end while
```

Similarly, we also use $\alpha_{\mathcal{L}}$ to denote the thermal characteristic of a subsequence. Let \mathcal{L} is a job sequence and $\mathcal{J}(\mathcal{L})$ denotes the jobs in \mathcal{L} . In our algorithm, a job sequence \mathcal{L} is regarded as a virtual job with the power consumption $\rho_{M,\mathcal{L}}$ and the execution time $W_{\mathcal{L}}$, which can be calculated as,

$$\rho_{M,\mathcal{L}} = \frac{\sum_{J_j \in \mathcal{J}(\mathcal{L})} E(W_j) \rho_{M,j}}{\sum_{J_j \in \mathcal{J}(\mathcal{L})} E(W_j)} \quad (14a)$$

and

$$W_{\mathcal{L}} = \sum_{J_j \in \mathcal{J}(\mathcal{L})} W_j \quad (14b)$$

Based on Equations(5,11, 12), $\alpha_{\mathcal{L}}$ for job sequence \mathcal{L} can be easily calculated.

4. SLACK ALLOCATION FOR TEMPERATURE REDUCTION

In this section, we introduce a slack allocation algorithm so that the randomness of workloads does not elevate the chip peak temperature in most cases. The Algorithm 1 schedules jobs based on the average execution time of jobs. On one hand, if job J_j is a hot job, and the execution time of its instance is longer than μ_j , then the new hot spot may be generated based on Algorithm 1; on the other hand, if job J_j is a cool job, and the execution time of its instance is shorter than μ_j , then the chip may not be able to effectively cool itself down by running J_j and the subsequent hot job can overheat the chip. However, to further slow down the cool job for cooling down the chip is not a good choice since the inherent reason that the temperature goes up is the hot job. So the case that the instance of the cool job runs less than its average execution time is regarded as the case that the instance of hot job runs more than its average execution time in a pair of hot-cool jobs.

We first deal with the case where the instance of the hot job runs more than its average execution time. We introduce a metric β_j^h which represents how much W_j of job J_j deviates above μ_j on average for the hot job. β_j^h is defined as

$$\beta_j^h = \sum_{w_{j,l} > \mu_j} \frac{(w_{j,l} - \mu_j)^2}{\sigma_j} p_{j,l} \quad (15)$$

where σ_j is the standard deviation of W_j .

Similarly, β_j^c is defined for the cool job to denote how much the execution time of the cool job J_j deviates below μ_j on average,

$$\beta_j^c = \sum_{w_{j,l} < \mu_j} \frac{(w_{j,l} - \mu_j)^2}{\sigma_j} p_{j,l} \quad (16)$$

Each hot job is paired with a cool job in Algorithm 1. Assume we have a pair of hot/cool jobs $\langle J_{2j-1}, J_{2j} \rangle$, where J_{2j-1} is a hot job and J_{2j} a cool job. β_{2j-1}^h is modified as following based on β_{2j}^c ,

$$\beta_{2j-1}^m = \beta_{2j-1}^h + \beta_{2j}^c \quad (17)$$

After the modified β_j^m for each hot job is calculated, the job slacks are distributed among all hot jobs based on each β_j^m . The slack allocated to hot job J_j is proportionate to β_j^m . Then the execution of job J_j is slowed down accordingly.

We define a *critical frequency* for each job. The critical frequency for job J_j is denoted as f_j^c , which states that if the initial temperature of the processor is T_{pref} , the temperature of the processor keeps unchanged after job J_j is executed at frequency f_j^c .

Algorithm 2 Slack Allocation Algorithm

Require: Job set $\mathcal{J} = \{J_j | 1 \leq j \leq N\}$

```

1: for  $j = 1, \dots, N$  do
2:   calculate  $\beta_j^m$  for each hot job
3: end for
4: sort jobs  $\{J_1, \dots, J_{N/2}\}$  based on  $\beta_j^m$ 
5: for  $j = 1, \dots, N/2$  do
6:   while slack available do
7:     assign slack to  $J_j$  proportional to  $\beta_j^m$ 
8:     slow down task  $J_j$ 
9:   end while
10: end for
```

Suppose $f_{i+1} \geq f_j^c \geq f_i$, then operating speed for job J_j should be not slower than f_i . If the slack for hot job J_j is not exhausted after its execution have been decreased to f_i , then the leftover

slack is exploited by other hot jobs instead of further slowing down J_j . The proposed algorithm is shown in Algorithm 2.

Note that available slack in line 6 in Algorithm 2 is the summation of individual task; the slack for single job is calculated from Equation 2. Also note that the available slack for J_j is upper bounded $slack_j$, calculated by Equation 2. If slack assigned to J_j is more than $slack_j$, conducted in line 7 in Algorithm 2, then the actual slack assigned to J_j is $slack_j$.

5. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we will first introduce the experiment setup and the benchmarks used. Then the experimental results are presented to evaluate the performance of the proposed algorithm.

5.1 Experimental Setup

We consider a DVFS-enabled processor similar to Intel’s Xscale processor [10] in the experiments. The processor has five operating frequencies: 150MHz, 400MHz, 600MHz, 800MHz and 1000MHz. Thermal parameters used in the experiments are the same as in [7]. The ambient temperature is set to $35^\circ C$ and the initial temperature set to $55^\circ C$. We choose $85^\circ C$ as the critical temperature of the chip and $80^\circ C$ as the preferred operating temperature as in [9].

Table 1: Benchmarks from Benchmark Suites

Benchmark Suite	Benchmark
MediaBench	jpegenc, jpegdec mpeg2enc, mpeg2dec
SPEC2000	applu, gcc, bzip2, crc32, mcf, mesa, swim

The application benchmarks are chosen from MediaBench and SPEC2000 benchmark suites, as shown in Table 1. Only the power-intensive applications impose the challenge to the thermal management; so workload type is critical to evaluate the performance of the proposed algorithm. In terms of the power dissipation, the benchmarks are grouped into three categories: hot (power-hungry applications), cool (power-non-hungry applications), and mild (power demand between power-hungry applications and power-non-hungry applications). In our experiments, the application consists of hot/mild/cool jobs. We consider the applications with the discrete distribution of execution time in our experiments. Applications with the continuous distribution of execution time could be similarly handled.

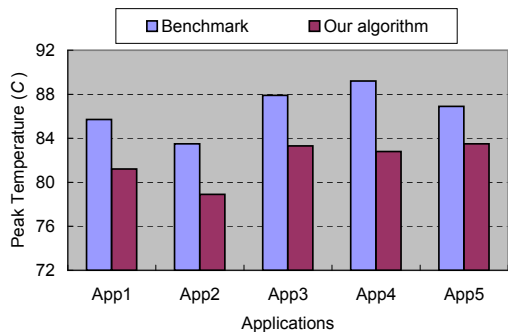


Figure 1: Peak Temperature Reduction

Based on benchmarks in Table 1, we construct applications used in our experiments. Applications with different number of jobs show the similar trends. We only report the results for five applications with each consisting of 8 jobs from Table 1. We assume each job has 5 choices of possible execution time ($0.01WCET$, $0.05WCET$, $0.1WCET$, $0.3WCET$, $WCET$), where $WCET$ is short for the Worst Case Execution Time. In order to evaluate the performance of the proposed algorithm, we also implement the approach in [8] as the benchmark.

We evaluate the effectiveness of our algorithm by comparing the peak temperature with our approach against the one with benchmark algorithm. The benchmark algorithm is implemented based on the average execution time of each task.

The simulation results are presented in Figure 1. From experimental results, we know that the proposed algorithm effectively reduces the peak temperature with when stochastic workloads. Our approach achieves a peak temperature reduction of as much as $6.4^\circ C$ and $4.7^\circ C$ on average. The reason is twofold: 1) the job sequencing helps reduce the peak temperature; 2) the hot jobs are slowed down and less heat is generated. Note that hot jobs are the rooted reason that chip will become overheated. Our algorithm only slows down the execution of hot jobs, which helps improve the thermal behavior of the processor by eradicating hot jobs. Hence, the limited task slack is smartly exploited to tackle the thermal related issue in our approach since cool jobs are not allowed to slow down its execution. In addition, the cool job which runs less than its average execution time helps the hot job in the same hot-cool job pair get more slack and then the hot job is stretched further. Accordingly, less heat is generated and the chip temperature goes down.

6. CONCLUSION

In this paper, we have proposed a thermal-aware scheduling algorithm for peak temperature reduction when the applications are the stochastic workload. The proposed approach interleaves the execution of the hot job and cool job so that the execution sequence of hot job and cool job can automatically better off the chip thermal behavior. On the other hand, the hot job is also stretched to further slow down the job execution and reduce the heat generation on the chip whenever job slack is available. The experimental results show that the proposed algorithm effectively reduces the thermal emergencies of the chip with applications where workloads are stochastic.

7. REFERENCES

- [1] N. Bansal and K. Pruhs, “Speed scaling to manage temperature,” in *Proc. of STACS*, pages 460-471, 2005.
- [2] Y. Liu, H. Yang, R.P. Dick, H. Wang, and L. Shang, “Thermal vs energy optimization for dvfs-enabled processors in embedded systems,” in *Proc. of ISQED*, 2007.
- [3] S. Liu, Q. Qiu and Q. Wu, “Full-chip leakage current estimation based on statistical sampling techniques,” in *Proc. of GLSVLSI*, 2008
- [4] S. Liu, Q. Qiu and Q. Wu, “A probabilistic technique for full-chip leakage estimation,” in *Proc. of ISLPED*, 2008
- [5] S. Zhang and K. S. Chatha, “Approximation algorithm for the temperature aware scheduling problem,” in *Proc. of ICCAD*, 2007.
- [6] J.R. Lorch and A. J. Smith, “Improving dynamic voltage scaling algorithms with pace,” in *Proc. of ACM SIGMETRICS*, 2001.
- [7] S. Zhang and K. S. Chatha, “System-level Thermal Aware Design of Applications with Uncertain Execution Time,” in *Proc. of ICCAD*, 2008.
- [8] R. Jayaseelan and T. Mitra, “Temperature aware Task Sequencing and Voltage Scaling,” in *Proc. of ICCAD*, 2008.
- [9] S. Liu, J. Zhang, Q. Wu, and Q. Qiu, “Thermal-Aware Job Allocation and Scheduling for Three Dimensional Chip Multiprocessor,” in *Proc. of ISQED*, 2010
- [10] “Intel-Xscale Micro-architecture,” available at <http://www.intel.com>
- [11] S. Gunther et al, “Managing the Impact of Increasing Microprocessor Power Consumption,” Intel Technology Journal, 2001
- [12] G. Casella, R. L. Berger, “Statistical Inference,” Dover Publications, 2003