# Thermal-Aware Sampling in Architectural Simulation

Ehsan K.Ardestani, Elnaz Ebrahimi, Gabriel Southern, and Jose Renau
Dept. of Computer Engineering
University of California Santa Cruz
{eka, elnaz, gsouther, renau}@soe.ucsc.edu

## ABSTRACT

Thermal behavior of modern processors is a first-order design constraint. However, accurate estimation of thermal behavior is time consuming, and techniques for accelerating performance simulations often yield inaccurate results when directly applied to thermal simulation, or do not reduce the thermal computation at all. This paper is the first to propose thermal sampling techniques. It can be integrated with existing phase-based and statistical-based architectural simulator sampling. The resulting simulator can perform accurate performance, power, and thermal characterization at close to 30 MIPS, on average, instead of 5 MIPS for the fastest sampling technique without thermal-aware sampling.

## Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems—*Modeling Techniques*

## 1. INTRODUCTION

The development of processors relies heavily on architectural simulators which allow researchers and designers to evaluate proposed ideas early in the design phase. However, these simulations are orders of magnitude slower than native execution and it is often necessary to use sampling techniques to reduce simulation time. These sampling techniques, which work well for performance-only simulation, are often slow and/or inaccurate for simulations that model thermal characteristics. The thermal characteristics of a processor affect multiple aspects of processor and system design, including maximum operating frequency, leakage power, performance throttling, and cooling solutions. As a result, without thermal simulation, it is not possible to obtain representative performance or power models.

Sampling solves the problem of slow simulation by reducing the number of instructions that need to be simulated when evaluating a proposed idea. Many studies have been done to evaluate how many instructions need to be simulated in order to accurately estimate performance. Phase-based sampling and statistical-based sampling are the two basic methods of reducing the number of instructions that need to be simulated while still gathering accurate performance statistics [1–5].


*ISLPED'12,* July 30–August 1, 2012, Redondo Beach, CA, USA.


Martínez *et al.* [6] show that these sampling methods do not work well when directly applied to thermal simulation. Coskun, *et al.* [7] developed a phase-based simulation methodology to apply sampling to performance and power computations. It provides accurate temperature results, but does not perform thermal sampling, which leaves thermal simulation as the main bottleneck. There is no accurate thermal simulation methodology based on statistical sampling. In this paper, we 1) extend sampling to the thermal domain for both phase-based and statistical sampling, and 2) for the first time develop an accurate statistical sampling based thermal simulation (*TASS*: Thermal-Aware Statistical Sampling). As a result, the thermal computation demand in the simulation is reduced by 25 times.

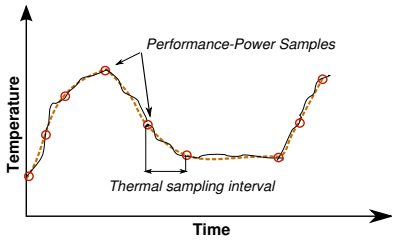


**Figure 1: Thermal sampling**

The key contribution of this paper is the proposal and development of practical implementations to perform thermal sampling. The proposed methodology leverages the slow state transition in thermal domain to address the sampling unfriendly feature of thermal domain: strong dependency on the previous states. Figure 1 shows a thermal profile for a generic application. The original thermal trace is shown with continuous lines. The performance and power sampling points are shown by circles. Each performance sample is typically thousands of instructions, while the sampling interval is tens of thousands to millions of instructions. To generate the temperature trace, previous works reconstruct the power trace and pass it to the thermal model. Consequently, with or without performance sampling, the number of thermal computations remains the same. While the previous works are oblivious to the sampling at performance stage, our proposed method integrates with it. The dashed line between each two adjacent circle in the figure is one thermal sampling interval. One round of thermal computation is enough in the proposed method to generate the temperature trace for a thermal sampling interval. This benefits the simulation in two connected ways; First, it makes it feasible to extend sampling to thermal by simplifying the integration of performance and thermal sampling. Second, it saves time by significantly reducing the number of thermal computations; We propose a framework to preserve the accuracy while applying this idea.

## 2. BACKGROUND

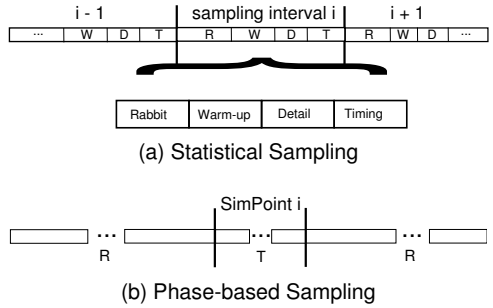


**Figure 2: Execution modes in sampling methods**

## 2.1 Statistical-based Sampling

When sampling is used with performance modeling the goal is to produce an accurate average estimate. Some structures like caches and branch predictors require warm-up when the samples are not long enough [4]. Figure 2a shows the sequence of execution in statistical sampling process. Each sampling interval is composed of 4 execution modes (or 3 as some methods do not implement *Rabbit* mode, all modes described in Table 1). The *Timing* is where the samples are gathered.

| Phase | Description |
|---|---|
| ***R***abbit | Fast-forward emulation or native co-execution |
| ***W***arm-up | Memory and branch traces to maintain accurate state |
| ***D***etail | Cycle-accurate modeling, statistics are discarded |
| ***T***iming | Cycle-accurate timing modeling |

**Table 1: Simulation execution modes.**

Statistical-based sampling encompasses power sampling as well, but is not commonly used for modeling thermal behavior. However, it has been used by Nookala, et al. for temperature-aware floorplanning [8]. They did not evaluate the overall accuracy of their sampling methodology, which was only used to help guide a floorplanning optimization algorithm. Our evaluation shows that even an enhanced Nookala methodology that considers CPI variations over time (which we call *SS*) does not yield accurate results as extra thermal-aware adjustments are required for sampling parameters (See Section 3). Dynamic adaptations in the architecture can be captured by statistical sampling at runtime.

## 2.2 Phase-based Sampling

When the samples are long enough, such as 100M instructions used in SimPoint [1], it is not necessary to perform microarchitectural warm-up to achieve accurate performance metrics. Hence, phase-based methods go through 2 execution modes, *Rabbit* and *Timing*. The samples, called SimPoints, are gathered during the *Timing* mode. Figure 2b shows the modes in the sampling process. The dotted lines in the figure indicates that the simulation points and the fast-forwarding intervals are much longer in phase-based sampling compared to the statistical sampling technique. Accordingly, the number of samples are much less.

Coskun, et al. [7] proposed reconstructing the power trace using SimPoints [1]. This is done in two simulation steps after a round of offline profiling. The first one gathers performance and dynamic power for the SimPoints. The second one reuses the generated dynamic power and performance results, and reconstructs the whole program execution. The program phases that were fast-forwarded are approximated with the most similar SimPoint. The second simulation step reconstructs the power trace and computes the temperature. This technique does not deploy thermal sampling, and requires modeling the temperature for the whole trace. We call this method *PS*.

The phase-based approach of obtaining power traces integrates very well with SimPoint when there is no need for processor adaptation based on temperature or power. However, whenever the processor performs any architectural adaptation based on power or temperature, its performance changes. Coskun, et al. [7] proposes generating the performance and power SimPoints for each possible processor state. For example, if there are 4 DVFS levels, the SimPoints need to be generated 4 times. The second simulation phase picks the correct trace to build the correct runtime power and thermal profile.

# 3. THERMAL-AWARE SAMPLING

The existing sampling based thermal simulations perform sampling, and then reconstruct the power trace. Afterwards, a full thermal computation is performed on the reconstructed power to generate a detailed temperature trace. Hence, with or without performance sampling, the thermal stage performs the same amount of computations. The slower state transitions in temperature compared to power and performance suggests the potential for having fewer thermal computations. Previous works have recognized this, and most of them average the power for 10K to 1M cycles. Then the thermal computation is performed once for that timestep [8–10]. The timestep is set to be around an order of magnitude smaller than the thermal time constant ($TC$) to provide accurate transients, avoiding thermal computations for every cycle. Nevertheless, while sampling methods drastically reduce the time spent on performance simulation, the execution time for thermal simulation remains a limiting factor to improve the simulation speed.

## 3.1 TASS Method

We extend sampling to the thermal stage. The main challenge is that unlike performance sampling, temperature sampling requires a very long warmup. In other words, temperature has a strong dependency on previous state, which is at odds with the idea of sampling. However, as mentioned earlier, temperature state changes much more slowly than power and performance. This is leveraged to avoid excessive thermal computations. We start explaining the methodology with the statistical sampling technique. A similar concept is applicable to the phase-based technique which will be discussed later.

In general, precise estimation of temperature at time $T_1 = t$ depends on correct estimation of three parameters:

- temperature at time $T_0 = t - \delta$,
- power consumption for the $(T_0 : T_1)$ interval, and
- the length of the interval, $\delta$.

For sampling, $\delta$ is the length of thermal interval. With longer $\delta$, more distribution information will be skipped. Therefore, $\delta$ can be adjusted for appropriate trade off between speed and accuracy. We set the length of thermal interval equal to the performance sampling intervals ($TPr = 1$, explained later in this section). Note that thermal intervals shorter than performance intervals do not provide more accurate temperature trace. The reason is that smaller thermal intervals than the sampling interval do not contain more power distribution information. This is something that previous work has neglected, and as a result they use a constant timestep which is independent of the performance sampling interval.

We synchronize the beginning of thermal and performance intervals. The only temperature value computed for the thermal interval is located at the end of the interval. Note that the correct estimation of temperature at $T_0$ itself is interdependent on the previous power and $\delta$ values. Therefore, the main challenge is the correct estimation of *Power* and *Time* for each interval. However, these values are only available for the samples ($T$) rather than the whole interval. We use a weighted moving average to estimate power and time for a thermal interval, as formulated in Equation 1.

$$\Theta_i = \frac{\sum_{k=1}^{n} \alpha_k \times \theta_{i-k}}{\sum_{k=1}^{n} \alpha_k}, \alpha_k = \frac{1}{2^k} \quad (1)$$

$\Theta$ and $\theta$ stand for *estimated* and *measured* value respectively. Note that the estimated value will be used as the representative value for a whole interval (a full *RWDT* sequence). The measured value is the value gathered only during the sampling ($T$ mode). $n$ is the *history size*, which determines how many measured samples are used in calculation of the estimated value for the current interval.

### 3.1.1 Thermal-aware filtering for power

By estimating power values for the fast forwarded points in intervals, the power trace is reconstructed. In our experiments, we set $n = 7$ in Equation 1. We use the *measured* power from intervals $i$ to $i-7$ to obtain an *estimated* power for the whole interval $i$. Averaging the power values also works as a filtering mechanism to smooth high frequency power spikes. We propose the filtering because power spikes in the samples affect the reconstructed trace, which can increase the error in the reconstructed temperature even more, given that temperature effects tend to stay longer due to the *TC* effect. Instead of applying a filter, we could increase the length of each sample (i.e. simulating more instructions for each sample). However, it would be at the cost of longer simulation time, as detailed timing simulation is slow. The simple filtering process relaxes the demand for longer sampling length, while still providing stable estimations. As the evaluation in Section 5 shows, the recommended performance-driven statistical-sampling parameters (*SS*) yields inaccurate results due to this spike sensitivity during reconstruction.

$$T_i = \sum_{k=1}^{i} \delta_k, \delta_k = \frac{Cyc_k}{ClkFreq}, Cyc_k = CPI_k \times Inst \quad (2)$$

### 3.1.2 Timeline Reconstruction

The length of sampling intervals are always a fixed number of instructions (Table 4). However, due to the time variant nature of the *CPI*, the actual length of each interval in terms of cycles varies. To estimate the length of each interval, Equation 1 is used with measured *CPI* values from last 7 intervals. Given the estimated *CPI* for interval $i$, we can estimate the number of cycles (*Cyc*) for the interval using Equation 2. The timeline of the execution is reconstructed by estimating the length of each interval ($\delta$). Eventually the time at the end of each interval ($T$) is estimated as formulated in Equation 2.

### 3.1.3 Sampling Parameters

SMARTS [4] specifies the length of each sample and the sampling intervals based on the observed variability in the program behavior (IPC). We use the same methodology. In addition, the coefficient of variability of the power samples can also be used to guide the parameter selection. Our observation is that longer samples need to be measured to provide stable temperature results. Table 4 shows the parameters.

Thermal phases are longer than performance phases [6]. This implies using even longer thermal intervals than performance intervals. We define the natural value $TPr \geq 1$ to be ratio of the thermal to performance sampling interval. For example, $TPr = 2$ means that each thermal interval is as long as 2 performance intervals in terms of number of instructions. We run a set of experiments with different thermal sampling intervals, while performance sampling intervals stay the same. The results are shown in Figure 3. The y-axis shows the Root Mean Square Error (RMSE) of comparing the resultant thermal traces against the shortest thermal interval of 10K. The results confirm that as long as the thermal statistics are gathered at the end of thermal interval, where the temperature is estimated, longer thermal sampling intervals can be tolerated. As a results, $TPr > 1$ can be used to even further accelerate the simulation speed. Practically $TPr = 3$ still generates accurate results. This means that the thermal interval is around 15M cycles, which is longer than thermal time constant of the package we model. This is the first time that a thermal simulation can run with a timestep longer than *TC* and preserve accuracy.

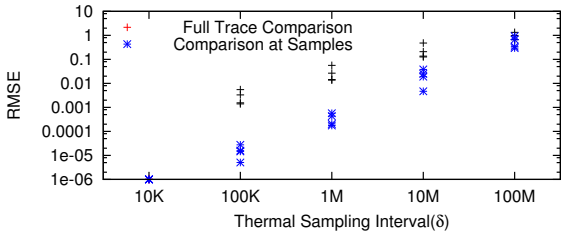


**Figure 3: Accuracy of estimated temperature at thermal samples with different thermal sampling intervals**

## 3.2 TAPS Method

As mentioned in Section 2.2, Coskun *et al.* [7] proposed a power reuse method to reconstruct the power model for phase-based methods. The distance between samples in phase-based sampling is longer and variable (each sample is typically in order of 10M to 100M instructions and the distance between samples could be billions of instructions). Martínez *et al.* [6] point to the validity of reporting the metrics only at samples as long as the initial temperature for each sample is set to the correct value. So fast-forwarding through the intermediate points and estimating the temperature for the end of the interval between samples (i.e. beginning of next sample) avoids multifold thermal computations, and speeds up the simulation as a result. For fast-forwarding, a faster but less accurate thermal simulation can be achieved by either performing thermal simulation on a coarser grain floorplan, or with a longer time step. We only perform the latter because this is enough to reduce the thermal simulation time to a small percentage of the total simulation time.

# 4. SIMULATION SETUP

Table 2 lists the methods we evaluate in this work. *SS* uses the same sampling parameters as suggested in [4]. Both *PS* and *SS* methods perform thermal computation after reconstructing power trace with 100K cycle timestep.

| Method | Description |
|---|---|
| Full | Full simulation, (no sampling) |
| PS | Phased-base sampling (Perf. + Power sampling) |
| TAPS | *PS* + (Thermal sampling) |
| SS | Statistical-based sampling (Perf. + Power sampling) |
| TASS | Thermal-aware SS, (Thermal sampling) |

**Table 2: Simulation methods evaluated in this work**

For performance simulation, we use a modified version of SESC [11] that uses QEMU [12] as the functional emulator executing SPARC instructions. The simulator offers 4 different execution modes, explained in Table 1, to support sampling. We configure SESC to pass activity counters to McPAT [13] (every 100K instructions max) which we use for calculating power. We use a modified version of SESCTherm [14] to scale leakage power consumption

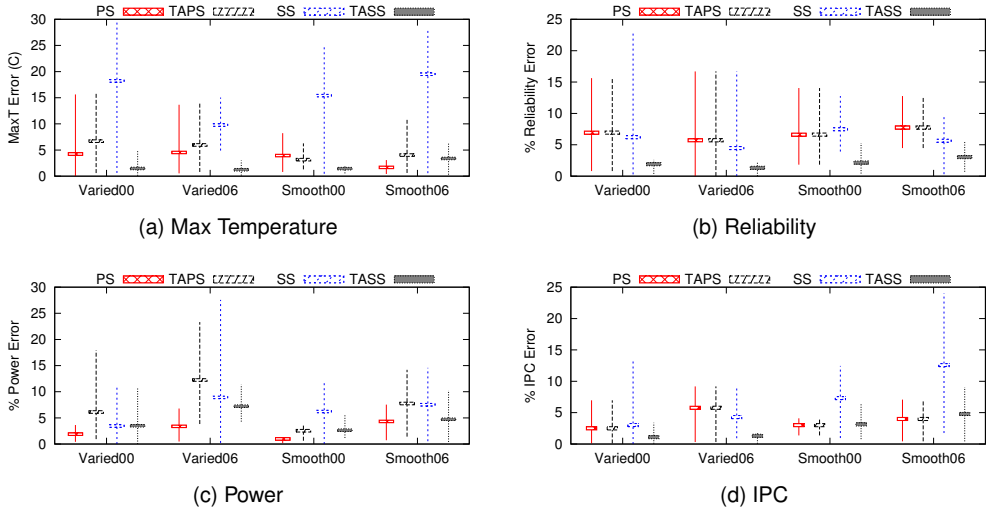


**Figure 4: Average, minimum, and max error of different methods. *TASS* outperforms the other methods. *SS* triggers the DTM method, and as a result has high IPC error, in addition to inaccuracy in estimating thermal metrics.**

according to temperature and device properties. We simulate two processors: one is an Intel Nehalem-like high performance (HP) core, and the other is an AMD Bobcat-like low power (LP) mobile processor (Table 3). Table 4 shows the sampling and the thermal simulation parameters respectively. For Dynamic Thermal Management (DTM), we implement thermal throttling. When the chip temperature exceeds a threshold (363K and 348K for HP and LP configurations respectively), the processor gates the clock to prevent physical damage until the temperature drops under the defined threshold.

| | Parameter | Value | | Parameter | Value |
|---|---|---|---|---|---|
| **HP** | Freq | 3.0 GHz | **LP** | Freq | 1.6 GHz |
| | I$ | 32KB 2w (2c hit) | | I$ | 32KB 2w (2c hit) |
| | D$ | 32KB 8w (3c hit) | | D$ | 32KB 2w (2c hit) |
| | L2 | 1MB 16w (12c hit) | | L2 | 512KB 4w (12c hit) |
| | Mem. | 180 cyc | | Mem. | 90 cyc |
| | BPred. | Hybrid 76Kb mem | | BPred. | Hybrid 38Kb mem |
| | Issue | 4 | | Issue | 2 |
| | ROB | 256 | | ROB | 56 |
| | IWin. | 48 | | IWin. | 20 |
| | Reg(I/F) | 128/128 | | Reg(I/F) | 80/64 |
| | Tech. | 32 *nm* | | Tech. | 32 *nm* |

**Table 3: Architectural parameters**

| Method | Parameter |
|---|---|
| PS | BBV = 1e8 |
| TAPS | MaxK = 10 [1] |
| SS | R = 0, W = 997e4, D = 2e4, T = 1e4 |
| TASS | R = 257e4, W = 250e4, D = 2e4, T = 7e4, History Size = 7, TPr = 2 unless mentioned otherwise |

**Table 4: Sampling parameters.**

The benchmarks that we selected from CPU2000 and CPU2006 suites are shown in Table 5. We also use the same set of metrics presented in [6] for the evaluation, namely maximum temperature, reliability (aggregated *EM*, *SM*, *TC*, *TDDB*, *NBTI*), and power. For power, we report the aggregation of leakage and dynamic power.

# 5. EVALUATION

| Suite | Category | benchmark |
|---|---|---|
| CPU2000 | Varied | swim, gcc, mesa, facerec, lucas, bzip2 |
| | Smooth | gzip, mgrid, applu, vpr, crafty, twolf |
| CPU2006 | Varied | gcc, milc, dealII, mcf |
| | Smooth | perlbench, soplex, astar, povray, namd, h264ref |

**Table 5: Selected Benchmarks**

For each method we evaluate its accuracy, simulation speed and the impact of thermal triggered adaptations on the overall simulation speed. Thermal throttling and its impact on performance is also evaluated. The results of *LP* and *HP* processor configurations are averaged together, because they are statistically similar.

## 5.1 Accuracy

We have clustered the benchmarks in categories shown in Table 5. The accuracy of each benchmark is calculated in comparison with *Full* and average result for each category is reported. We also report the minimum and maximum error for each category.

For *SS*, the spikes adversely affect the instantaneous metrics such as *maxT*. *TASS* generates accurate *maxT* results which is indicative of the effectiveness of thermal-aware power filtering and reconstruction. The average metrics like *Power* or *Reliability* are less sensitive to spikes as expected. The *SS* results in very high IPC error. The reason is that temperature spikes trigger the thermal throttling DTM mechanism. Thermal throttling gates the processor clock to allow for it to cool down. This has performance impact. When we disable the thermal throttling, the IPC error of *SS* and *TASS* are similar. On average, *TAPS* reaches the same order of accuracy as *PS*, 6.7% vs. 4.7%. Both *TAPS* and *PS* methods could benefit from extracting shorter SimPoints, *e.g.*, 10M instruction. However, we only run the experiments with the recommended configuration. On average, across all the benchmarks and metrics, *TASS* outperforms other evaluated methods in terms of accuracy with 3.6% average error.

We also compute the breakdown of accuracy of the mean temperature across the chip for two different Confidence Intervals (CI) of 95% and 99.7%. The results show that even choosing a confidence level of 99.7%, the average CI equals $\pm$ 0.17 for mean of 40.50 (Max CI = 1.02 for gcc06, mean 38.79). This means that

with 99.7% confidence that the result has less than 1C difference even for the worst case application.

## 5.2 Simulation Speed

The sampling methods are intended to minimize slow *D* and *T* detailed and timing simulation modes, and maximize the use of fast *W* and *R* modes (see Table 1). The thermal-aware methods also try to minimize the number of calls to the thermal model. The last column in Table 6 shows the simulation speed for each method on average across all the benchmarks. *TASS* has a maximum speed of 30 MIPS and an average speed of 18 MIPS, while *TAPS* has a maximum of 50 MIPS and an average of 30 MIPS. *SS* and *PS* are slower as they do not deploy thermal sampling and fast-forwarding. Also the spikes in *SS* trigger DTM, and lowers the simulation speed. *Full* runs at 0.76 *MIPS* on average. These results are generated with $TPR = 2$ (explained in Section 3) for *TASS*, which translates to thermal fast-forwarding of 10M instructions. For *TAPS* there are two different fast-forwarding rates as the samples are much longer and placed farther from each other. Within the samples, the fast-forward is set to 4M instructions, and for the intermediate points between the samples it is set to 40M.

| Method | Speed w/o thermal | | | | Speed | Speed |
|---|---|---|---|---|---|---|
| | R | W | D | T | w/o thermal | w/ thermal |
| Full | - | - | - | 1.0 | 1.1 | 0.8 |
| PS | 62 | - | - | 1.0 | 38 | 4.1 |
| TAPS | 62 | - | - | 1.0 | 38 | 30 |
| SS | - | 49 | 0.4 | 0.3 | 44 | 2.8 |
| TASS | 20 | 19 | 0.9 | 1.0 | 22.5 | 18 |

**Table 6: Breakdown of simulation speed in MIPS with and without thermal computations**

Extending sampling to thermal stage accelerates simulation speed around 7 times, while the accuracy degrades from 4.7% to 6.7% on average. It also affects the maximum error with the same rate. Table 6 shows the breakdown of execution time for each simulation mode, as well as the simulation speed with and without thermal sampling. The execution time with thermal for thermal-aware methods (*TAPS* and *TASS*) is an order of magnitude less compared to their thermal-unaware counterparts (*PS* and *SS*). *SS* without thermal is faster than *TASS* as the sampling parameter for *TASS* is adjusted for thermal accuracy. The execution time at different stages of simulation is distributed more or less evenly in the same order with thermal time now being around 25% of total execution time for both methods. Longer continuous instruction in each mode provides a better opportunity for QEMU to optimize the functional simulation and chain basic blocks.

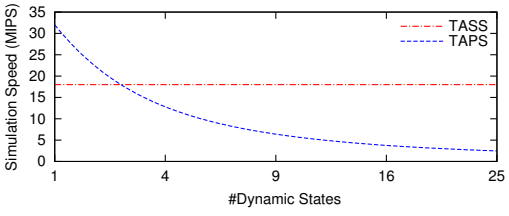


**Figure 5: TAPS requires more simulation as processor adapts dynamically, while TASS is insensitive to the dynamic adaptations.**

Our experiments show that modeling a system with the number of dynamic architectural states equal or greater than 4, *TASS* achieves faster speed than *TAPS*. In such a case, *TASS* is both faster and more accurate. Figure 5 shows the trend of speed regarding the number of dynamic reconfigurations.

## 5.3 Full Trace Reconstruction

Thus far, we reported the results with the thermal sampling tuned for speed ($TPr = 2$), which have accurate results at thermal samples. We also tune the parameters to have a fully accurate reconstructed thermal trace, which is important to the *time − varient* category of studies, e.g. evaluation of Dynamic Thermal Management (DTM) methods. We set $TPr = 1$ for *TASS* to have the same performance and thermal sampling intervals. For TAPS, we apply the same 4M fast-forwarding for all the samples. This results in an average error of 5.7%, while the speed is 14 and 23 *MIPS* for *TASS* and *TAPS* respectively. Figure 6 shows the accuracy of each method, and Figure 7 depicts the thermal trace for four benchmarks.

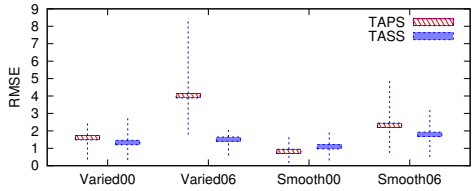


**Figure 6: Average, minimum and maximum RMSE across all benchmark categories.**

## 5.4 Thermal Throttling Effect

In addition to thermal behavior, throttling impacts performance as well. Table 7 shows the IPC with and without thermal throttling. It also provides the percentage of time that the processor spends in throttling. For *TASS*, the throttling time is fed back to the performance simulator to calculate the impact on IPC. This is possible because *TASS* does both performance and thermal simulation in one pass.

| Benchmark | IPC | Throttled IPC | | | % Time in Throt | | |
|---|---|---|---|---|---|---|---|
| | | Full | TAPS | TASS | Full | TAPS | TASS |
| gzip | 0.99 | 0.99 | 1.01 | 0.99 | 0 | 0 | 0 |
| swim | 0.45 | 0.45 | 0.42 | 0.45 | 0.09 | 0.07 | 0.06 |
| mgrid | 0.75 | 0.72 | 0.68 | 0.75 | 4.45 | 7.26 | 5.9 |
| applu | 0.76 | 0.75 | 0.72 | 0.75 | 1.30 | 0.82 | 1.70 |
| vpr | 0.87 | 0.87 | 0.80 | 0.85 | 0 | 0 | 0 |
| gcc00 | 1.29 | 1.12 | 1.71 | 1.21 | 13.1 | 17.7 | 12.4 |
| mesa | 1.50 | 1.35 | 1.17 | 1.40 | 9.87 | 14.8 | 11.9 |
| crafty | 1.51 | 1.34 | 1.21 | 1.40 | 11.3 | 13.8 | 10.0 |
| facerec | 0.72 | 0.72 | 0.72 | 0.69 | 0 | 0.98 | 3.57 |
| lucas | 0.65 | 0.65 | 0.63 | 0.64 | 0.10 | 0 | 0.12 |
| bzip2 | 1.06 | 1.03 | 1.09 | 1.02 | 2.69 | 4.26 | 3.25 |
| twolf | 0.88 | 0.88 | 0.84 | 0.81 | 0 | 0 | 0 |
| perlbench | 1.29 | 1.20 | 1.16 | 1.20 | 7.32 | 17.1 | 7.57 |
| gcc06 | 1.00 | 0.93 | 0.73 | 0.94 | 6.86 | 5.82 | 7.61 |
| mcf | 0.33 | 0.33 | 0.37 | 0.32 | 0 | 0 | 0 |
| milc | 0.50 | 0.49 | 0.43 | 0.50 | 2.39 | 4.61 | 3.01 |
| namd | 1.51 | 1.24 | 1.09 | 1.26 | 17.89 | 18.9 | 18.8 |
| dealII | 1.12 | 1.11 | 0.98 | 1.11 | 0.68 | 1.36 | 1.08 |
| soplex | 0.42 | 0.42 | 0.37 | 0.41 | 0 | 0 | 0 |
| povray | 1.88 | 1.56 | 1.39 | 1.48 | 16.9 | 21.8 | 18.1 |
| h264ref | 1.34 | 1.24 | 1.06 | 1.25 | 7.44 | 9.85 | 8.76 |
| astar | 0.62 | 0.62 | 0.43 | 0.59 | 0.04 | 0.14 | 0.05 |

**Table 7: Impact of thermal throttling on performance.**

# 6. CONCLUSION

This paper shows the necessity of applying sampling to thermal simulation in order to increase productivity by accelerating thermal simulation in architecture level. It proposes a thermal sampling method that is applicable to both statistical and phase-based sampling techniques. The evaluation shows that the newly proposed thermal-aware sampling methods, *TASS* and *TAPS*, outperform existing thermally-unaware methods to estimate performance, power,

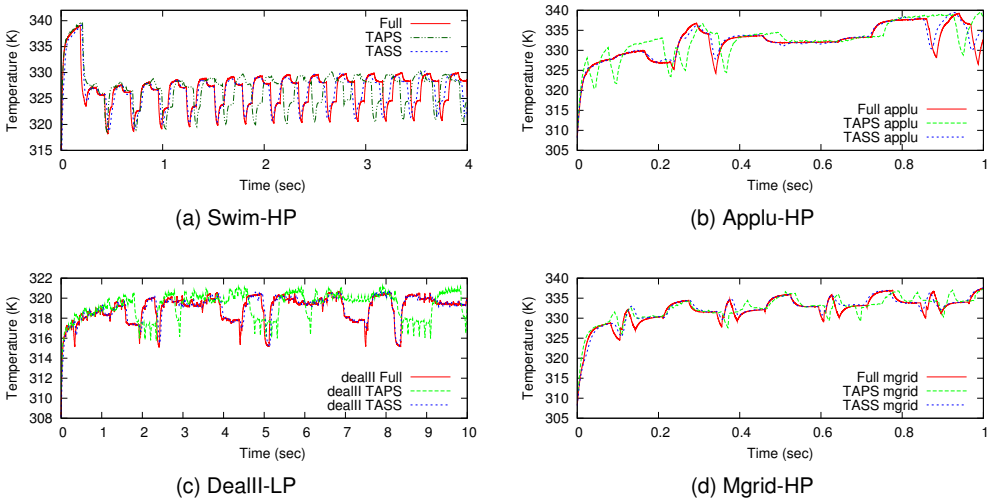


**Figure 7:** *TAPS* **reuses similar simpoints for the intervals with no sample, and does not follow the trend as tightly as** *TASS*.

and thermal characteristics, achieving a simulation speeds of 18 and 30 MIPS respectively on average. Without thermal-aware sampling simulation speed is less than 5 MIPS. With thermal-aware sampling the thermal simulation overhead is reduced by 25 times. *TASS* also has advantages in addition to fast simulations. First, it is more accurate than *PS*, *SS*, and *TAPS*. Second, the simulation speed is independent of the number of dynamic states of system (e.g number of DVFS states). We plan to make our simulation infrastructure available for other researchers to use.

## 7. ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation under grant 1059442; Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the NSF.

## 8. REFERENCES

[1] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically characterizing large scale program behavior," in *Proceedings of the 10th international conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Oct. 2002, pp. 45–57.

[2] W. Liu and M. Huang, "EXPERT: Expedited Simulation Exploiting Program BehaviorRepetition," in *International Conference on Supercomputing*, St. Malo, France, Jun–Jul 2004, pp. 126–135.

[3] D.G. Perez, H. Berry, and O. Temam, "Budgeted region sampling (beers): do not separate sampling from warm-up, and then spend wisely your simulation budget," in *Signal Processing and Information Technology, 2005. Proceedings of the Fifth IEEE International Symposium on*, Dec 2005, pp. 1 –6.

[4] R. E. Wunderlich, T. F. Wenisch, B. Falsafi, and J. C. Hoe, "Smarts: accelerating microarchitecture simulation via rigorous statistical sampling," in *Proc. 30th Annual Int Computer Architecture Symp*, 2003, pp. 84–95.

[5] Z. Yu, H. Jin, J. Chen, and L.K. John, "Tss: Applying two-stage sampling in micro-architecture simulations," in *International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS)*, Sep 2009, pp. 1 –9.

[6] F. J. M.-Martinez, E. K.Ardestani, and J. Renau, "Characterizing processor thermal behavior," in *Proceedings of the 15th international conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2010, pp. 193–204.

[7] A.K. Coskun, R. Strong, D. Tullsen, and T. Rosing, "Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors," in *Proceeding of the 11th International joint Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, Jun. 2009, pp. 169–180.

[8] V. Nookala, D.J. Lilja, and S.S. Sapatnekar, "Temperature-aware floorplanning of microarchitecture blocks with ipc-power dependence modeling and transient analysis," in *Proceedings of the 2006 International Symposium on Low Power Electronics and Design (ISLPED)*, Oct 2006, pp. 298–303.

[9] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," in *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*, June 2000, pp. 83–94.

[10] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-Aware Microarchitecture," in *Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA)*, Jun 2003, pp. 2–13.

[11] J. Renau, F. Basilio, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, S. Sarangi, P. Sack, K. Strauss, and P. Montesinos, "SESC simulator," January 2005, http://sesc.sourceforge.net.

[12] F. Bellard, "Qemu, a fast and portable dynamic translator," in *Proceedings of the annual conference on USENIX Annual Technical Conference*, Berkeley, CA, USA, 2005, ATEC '05, pp. 41–41, USENIX Association.

[13] S. Li, A. Jung Ho, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," in *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2009, pp. 469–480.

[14] J. N.-Battilana and J. Renau, "Soi, interconnect, package, and mainboard thermal characterization," in *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design (ISLPED)*, 2009, pp. 327–330.