A Quantitative Evaluation of the RAPL Power Control System

Huazhe Zhang and Henry Hoffmann
University of Chicago, Chicago USA
{huazhe,hankhoffmann}@cs.uchicago.edu

ABSTRACT

We evaluate Intel's RAPL power control system, which allows users to set a power limit and then tunes processor behavior to respect that limit. We evaluate RAPL by setting power limits and running a number of standard benchmarks. We quantify RAPL along five metrics: stability, accuracy, settling time, overshoot, and efficiency. The first four are standard measures for evaluating control systems. The last recognizes that any power control approach should deliver the highest possible performance achievable within the power limit. Our results show that RAPL performs well on the four standard metrics, but some benchmarks fail to achieve maximum performance. At high power limits, the average performance is within 90% of optimal. At middle power limits, it is 86% of optimal. At low power limits, the average performance is less than 65% of optimal.

1. INTRODUCTION

Processor designs are increasingly constrained by power and thermal dissipation. As a result, several *power control systems* have been proposed to guarantee system power consumption operates within a strict *limit* [5, 6, 11, 22, 30–32, 40]. In fact, Intel now supports power control directly in hardware through their Running Average Power Limit (RAPL) interface [7], which allows software to set a power limit that hardware ensures.

Any power control system takes a *power limit* as input and tunes behavior to ensure that this operating limit is respected. There are several desirable properties for any control system acting on a computer, which we illustrate in Fig. 1. These properties (identified by Hellerstein et al. [13]) allow us to quantify the behavior the control system over time and they include:

- **Stability:** freedom from oscillation.
- **Accuracy:** convergence to the limit.
- **Settling time:** duration until limit is reached.
- **Maximum Overshoot:** the maximum difference between the power limit and the measured power.

Collectively, these are referred to as the *SASO* properties. A power control system should be stable to avoid power (and thus thermal) fluctuations. The controller should be accurate to ensure that the system does not operate above the specified limit. It should have low settling time and low maximum overshoot to ensure that the power limit is reached quickly with only small errors.

In addition to these four properties (which are desirable for any control system), a power controller must also be *efficient*. That is, it should not only ensure operation within the power limit, it should also deliver the maximum possible performance subject to this power constraint.

This problem of maximizing performance within a power limit is especially important given the emergence of *dark silicon* [10, 37]. Dark silicon refers to the fact that modern processors cannot run at full speed without producing unsustainable power dissipation. Thus, an efficient power control system will both ensure that power limits are respected and do so by selecting which of the over-

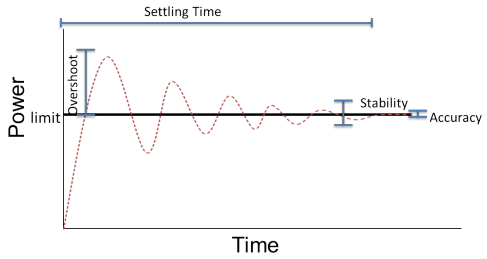

Figure 1: Illustration of the SASO properties.

abundance of transistors should be powered and at what speed.

This paper quantifies the behavior of the RAPL power control system. We run a number of benchmarks under a number of power limits on a Linux/x86 system based on Xeon E5-2690 processors. For each benchmark and limit, we record the achieved performance and power consumption. This data allows us to quantify stability, accuracy, settling time, maximum overshoot, and efficiency. In general, we find RAPL to behave well in terms of the SASO properties. It is generally stable and accurate with low settling time. We find that almost every benchmark and power target exhibits considerable overshoot, although the potential impact is limited due to the low settling time. We find RAPL's efficiency to be very sensitive to both the power target and the application under control. Very low power limits produce an average performance of just over 65% of optimal. Mid-level power caps produce an average performance of 86% of optimal. High power limits produce average performance close to 95% of optimal. However, for some applications, performance may be much lower than average for all limits.

This paper makes the following contributions:

- It proposes using the existing SASO properties to quantify the behavior of power control systems for processors.
- It proposes adding a fifth property, efficiency, to the study of processor power control to quantify the controller's ability to deliver maximum performance for a given power budget.
- It quantifies RAPL's behavior for these five properties.
- For each property evaluated, it states some implications of our findings for RAPL users.

The remainder of the paper is organized as follows. Section 2 provides further detail on the desired properties and our method for quantifying them. Section 3 describes the system and benchmarks used in our study. Section 4 presents our findings and describes their implications. Section 5 discusses related work in power management. The paper concludes in Section 6.

2. DESIRED PROPERTIES

RAPL is a *control system*. Its input is a power limit. Given this input, RAPL tunes system behavior to ensure the limit is respected. As a control system, RAPL is responsible for reacting to changes in system dynamics (*e.g.,* the application entering a new phase) to maintain the power target over time.

We quantify RAPL's performance in terms of the standard prop-

erties one would expect from a control system acting on a computer system. These properties are defined by Hellerstein et al. [13] and illustrated in Fig. 1: *stability, accuracy, settling time, and maximum overshoot*. Together, these four properties quantify the control system's ability to achieve the desired power consumption quickly and without error. We discuss these further in Section 2.1.

While the SASO properties quantify RAPL's temporal behavior in the power dimension, they fail to account for one additional, and essential, property of a power control system: *efficiency*. Efficiency refers to the controller's ability to deliver performance while respecting the power budget. The more efficient the control system, the higher the performance it will deliver for a given power limit. This property is discussed in more detail in Section 2.2.

2.1 Definition of SASO Properties

The SASO properties describe the behavior of a control system over time. As we are describing a computer system, we adopt a discrete time model. We assume a given power limit p_ℓ, and a measured power p_m. We denote the measured power at time k as $p_m(k)$. We assume that we have power measurements for n distinct times; *i.e.,* $k \in 1,\ldots n$.

Stability A *stable* system converges to a single value; *i.e.,* the derivative of the measurements becomes zero. The remaining SASO properties are defined in terms of a stable system. We quantify the stability of a power controller by computing the *standard deviation* of the power measurements over time:

$$stdev = \sqrt{\frac{1}{n}\sum_{k=1}^{n}[p_m(k)-\bar{p}_m]^2} \quad (1)$$

where $\bar{p}_m$ is the average measured power; *i.e.,* $\bar{p}_m = \frac{1}{n}\sum_{k=1}^{n} p_m(k)$. Low standard deviation indicates high stability.

Accuracy An accurate power controller converges to the power limit (or below). Control accuracy may be the most important property for users, because accurate control means that the limit is respected. For example, consider a power limit p_ℓ. A power controller is accurate if there exists a time k_{ss} (for steady-state) such that the measured power p_m converges to p_ℓ for all $k > k_{ss}$. Given power measurements over time, we quantify accuracy by computing the *mean absolute percentage error* (MAPE):

$$MAPE = \frac{1}{n}\sum_{k=1}^{n}\begin{cases} p_m(k) > p_\ell: & \left|\frac{p_m(k)-p_\ell}{p_m(k)}\right| \cdot 100\% \\ p_m(k) \le p_\ell: & 0 \end{cases} \quad (2)$$

Low MAPE indicates accurate control, while high MAPE represents inaccuracy. MAPE expresses error as a percentage of the power target, allowing comparisons across different targets.

Settling Time Settling time refers to the time that passes from when the limit is set (and control begins) to the point where the system becomes stable. That is, settling time is quantified as the difference between the start time k_0 and k_{ss}:

$$settle = k_{ss} - k_0 \quad (3)$$

Low settling times indicate the desired power is reached quickly.

Max Overshoot The maximum overshoot refers to the largest amount by which the system exceeds the power limit on its way to becoming stable. We quantify maximum overshoot as:

$$maxover = \max_k p_m(k) - p_\ell \quad (4)$$

Lower overshoots are better.

2.2 Efficiency

Efficiency represents the performance delivered within a power limit. Ideally, the power control system should deliver the highest possible performance for a given limit. We quantify efficiency by considering all configurations c of a particular processor, where each configuration represents a particular resource usage. For example, on a processor with two cores and two clock speeds (both cores must use the same speed), there are four configurations, each of which may deliver a separate power and performance. We denote the set of possible configurations as $C = \{0\ldots n_C - 1\}$, where n_C is the total number of configurations.

We quantify efficiency in terms of a particular workload. We first measure that workload's performance r_c and power consumption p_c in every possible configuration $c \in C$. We then set a power limit p_ℓ and measure the delivered performance at that target r_ℓ. With these values, we compute efficiency as:

$$\begin{aligned} Under &= \{c | c \in C \wedge p_c \le p_\ell\} \\ efficiency &= \frac{r_c}{\max_{c \in Under} r_c} \end{aligned} \quad (5)$$

The first equation creates a set $Under$ of all configurations with power consumptions below the limit. The second equation simply divides the measured performance under the control system by the largest performance of any configuration in the set $Under$. The higher the efficiency the closer the power manager is to delivering the best possible performance for a given limit.

3. EXPERIMENTAL SETUP

3.1 System

We use a dual-socket Intel/Linux system with a SuperMICRO X9DRL-iF motherboard and two Xeon E5-2690 processors. This motherboard supports setting software power limits through RAPL. The system runs Linux 3.2.0. We make use of the `msr` module, allowing access to the model specific registers (MSR) used to set RAPL power limits and read energy consumption. We use the `cpufrequtils` package to set the processor's clock speed. These processors have eight cores, fifteen DVFS settings (from 1.2 – 2.9 GHz), hyper-threading, and TurboBoost. In addition, each chip has its own memory controller, and we use the `numactl` library to control access to memory controllers. In total, the system supports 1024 user-accessible configurations, each with its own power/performance tradeoffs[1]. According to Intel's documentation, the thermal design power for these processors is 135 Watts.

3.2 Applications

We use 16 benchmark applications from three different suites including PARSEC (`x264`, `swaptions`, `vips`, `fluidanimate`, `blackscholes`, `bodytrack`) [2], Minebench (`ScalParC`, `kmeans`, `HOP`, `PLSA`) [29], and Rodinia (`cfd`, `nn`, `lud`, `particlefilter`)[4]. We also use a partial differential equation solver (`jacobi`) and the `swish++` search webserver [16]. These benchmarks test a range of important modern multicore applications with both compute-intensive and data-intensive workloads. All applications run with up to 32 threads (the maximum supported in hardware on our test machine). In addition, all workloads are long running, taking at least 10 seconds to complete. This duration gives us plenty of time to take measurements of system behavior.

[1] 16 cores, 2 hyperthreads, 2 memory controllers, and 16 speed settings (15 DVFS settings plus TurboBoost)

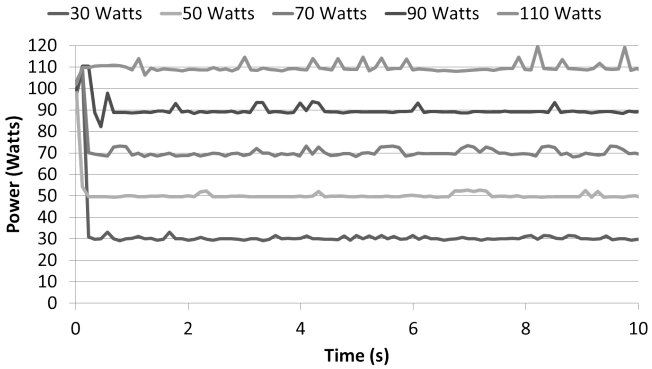

Figure 2: Example of measurements taken to determine SASO.

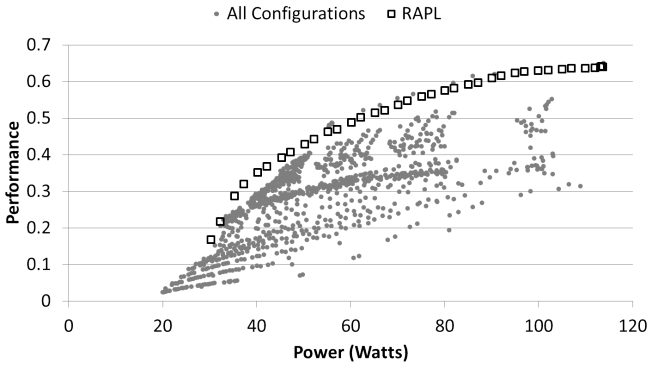

Figure 3: Example of measurements taken to determine efficiency.

3.3 Measurement

We measure performance as *throughput* – the rate at which an application completes. Higher numbers are better. Ideally, throughput would increase with increasing power consumption.

To evaluate the SASO properties, we measure power by collecting energy data by reading the appropriate MSR at 10ms intervals. We convert the energy measurement into a power consumption. At this point we have a series of power measurements at discrete times, and we use this data to calculate stability, accuracy, settling time, and max overshoot as described in Section 2.

To evaluate efficiency, we collect the average power consumption and performance data for each benchmark in each of the 1024 possible configurations of the system. Then, we run each benchmark while setting RAPL's power limit. We use 26 different RAPL power limits from 30W to 110W in 2.5W increments and collect the average power and performance data for each run. From this data, we directly calculate efficiency using Eqn. 5.

We use this measurement methodology to evaluate RAPL across a number of power limits. Specifically, we evaluate five power limits including: 30/50/70/90/110 Watts per processor. This allows us to test RAPL's ability to deliver both the SASO and efficiency properties across a wide range of power goals.

We illustrate the results of the measurement process in Figures 2–3. These figures show the measurements taken for the `jacobi` benchmark application. Fig. 2 shows the time series data we collect to evaluate the SASO properties. Time is displayed on the x-axis and the measured power consumption is on the y-axis. There are five curves corresponding to the five power limits we evaluate. Fig. 3 shows the power and performance measurements taken for `jacobi`, with power on the x-axis and performance (measured in throughput) on the y-axis. The small gray dots show points measured for each of the 1024 user-adjustable configurations. The boxes show measurements taken with RAPL. This benchmark represents a good case for RAPL, after some initial settling time, RAPL keeps power consumption close to the limits. Additional, the RAPL measurements for power and performance are close to the Pareto-optimal frontier achievable with user-adjustable configurations.

4. EVALUATION

In this section, we evaluate RAPL's SASO (stability, accuracy, settling time, maximum overshoot) and efficiency. For each one of these five properties, we offer comparisons between all 16 benchmarks. Out of the results, we find RAPL serves well in SASO, except for few exceptions discussed below. However, RAPL fails to deliver maximum efficiency for some benchmarks and power limits. For each of the five properties, we present our results and then discuss their implications.

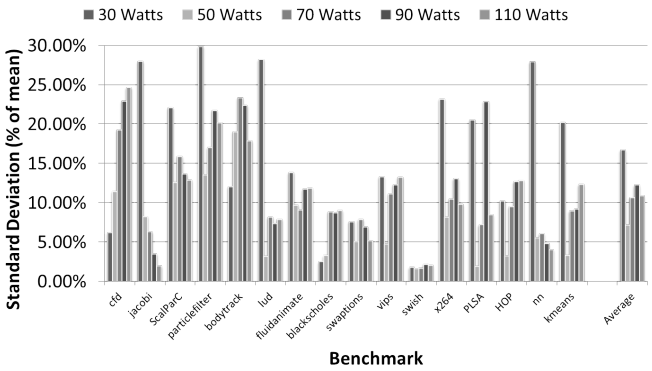

Figure 4: Stability quantified as standard deviation (lower is better).

4.1 Stability

We calculate stability according to Eqn. 1 and display the results in Fig. 4. This figure shows a somewhat high standard deviations. On average, all power limits have standard deviations over 5% of the mean, while all but the 50 Watt limit are over 10% of the mean.

There are two causes of instability. The first comes from large deviations beyond the limit prior to later becoming stable. This first cause is illustrated in Fig. 2 which shows that RAPL first overshoots low limits by a large amount before then stabilizing within a small window around the limit. The second cause of instability is inherently unstable applications. Applications in this class alternate between different phases, during highly parallel phases, the cpus are fully utilized, and RAPL meets the power limit. During serial phases, a smaller number of cpus are used and most are in idle state. Therefore in this phase, the power consumption is much lower than power limit. Fig. 5 shows an example of this second type of application which is inherently unstable.

RAPL handles the first case. The instability that may exist during start up is amortized by long running applications. We believe that most of our benchmarks are stable and stability would increase (standard deviation decrease) as the application runtime increases. Only two of our benchmarks appear to be inherently unstable: `bodytrack` and `particlefilter`. Our conclusion is that RAPL delivers overall good stability, but it does not stabilize inherently unstable benchmarks[2]. This last observation also implies that RAPL will not provide stability for systems that run many, short-lived tasks, and thus, constantly oscillate between high and

[2]We stress that this should not be viewed as a shortcoming of RAPL, but it is important for RAPL users to be aware of this fact.

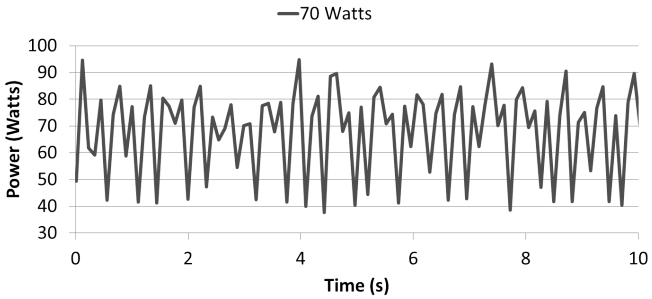

Figure 5: The inherently unstable `bodytrack` application. To increase readability, we show the measurements only for the 70 Watt limit. The other limits exhibit similar oscillations.

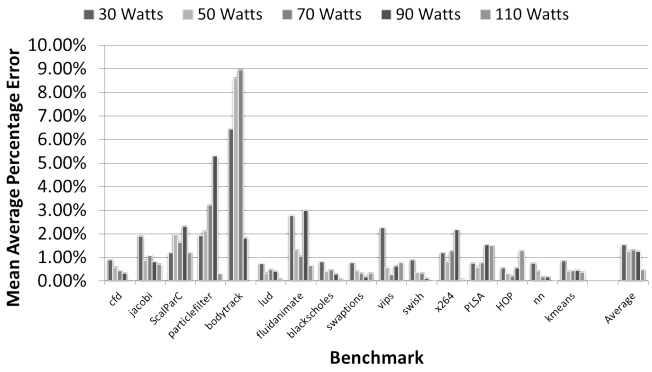

Figure 6: Accuracy quantified as MAPE (lower is better).

low CPU utilization.

4.2 Accuracy

We evaluate accuracy by calculating MAPE for each benchmark and power limit according to Eqn. 2. Fig. 6 shows the results. Fourteen of benchmarks have an error under 3% and two out of sixteen have errors over 5%. The benchmarks with bad accuracy are the two inherently unstable benchmarks: `bodytrack` and `particlefilter`. The inaccuracy for these two applications arises from their instability. Each time an application changes phase, RAPL reacts. When the benchmark transitions from low utilization to high utilization, RAPL over-allocates resources causing the power bound to be temporarily violated. Given frequent phase changes these overshoots of the power limit are also frequent. This dynamic is illustrated in Fig. 5, which exhibits frequent overshoots of the 70 Watt power limit for the `bodytrack` benchmark.

4.3 Settling Time

We evaluate settling time by computing Eqn. 3. We consider the steady state to be reached the first time the measured power is within 5% of the limit. The results are shown in Fig. 7. Overall, RAPL's settling time is low; all times are less than 0.9s and on average below 0.5s, which is quite short compared to the entire running time for our benchmark applications. These short settling times will matter more, however, for extremely short-lived applications.

4.4 Maximum Overshoot

We evaluate maximum overshoot by computing Eqn. 4 for each benchmark and each power limit, with the results shown in Fig. 8. In general, the lower the power limit, the higher the possible overshoot could be (since physical constraints limit maximum power consumption). However, due to the short setting time, the effect of

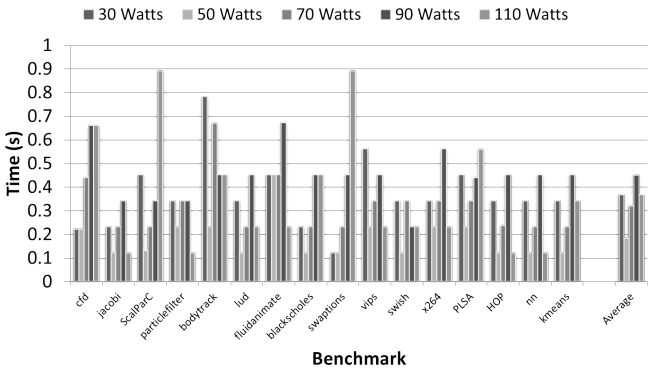

Figure 7: Settling time quantified in seconds (lower is better).

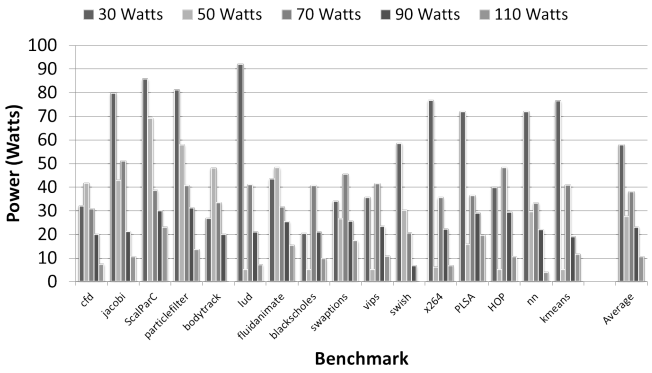

Figure 8: Maximum overshoot measured in Watts (lower is better).

large overshoot on the overall power consumption is still small.

4.5 Efficiency

This section evaluates RAPL's ability to deliver performance within a power budget. For each benchmark we measure performance and power consumption in each configuration of the machine (without RAPL). We then set the power limits and measure power and performance with RAPL. Given these measurements, we calculate efficiency according to Eqn. 5. We regard efficiency as the second most important property of a power control system (behind only accuracy) as users will want to know that they are achieving maximum performance within a power limit.

The results are displayed in Fig. 9. The results demonstrate that the delivered efficiency is sensitive to both the power limit and the application under control. For the 30, 50, and 70 Watt limits, the average efficiency is less than 0.9. For the 90 and 110 Watt limits, the average efficiency is slightly over 0.9. At the 30 Watt limit, no benchmark achieves an efficiency of greater than 0.85. At the 50 and 70 Watt limits, only seven benchmarks achieve efficiencies of greater than 0.9. For the 90 Watt limit, nine benchmarks are above the 0.9 efficiency threshold, while eleven of the sixteen surpass this efficiency at the 110 Watt limit.

For higher power limits, many applications achieve high efficiency. These applications all appear to be compute-intensive, in the sense that their performance scales nearly linearly with increased compute resources. The applications for which RAPL does not perform well at high power limits appear to be more communication-intensive. These applications have limitations in at least one resource. For example, the `x264` benchmark does not perform well when using hyperthreads on our test machine. It is possible to achieve slightly higher performance for the same power limit by disabling hyperthreads for `x264`, but RAPL has no control

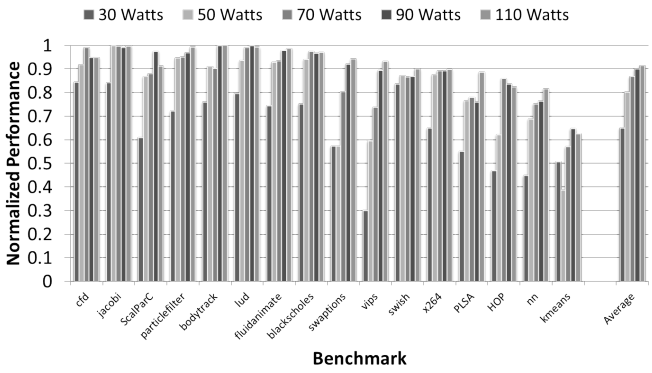

Figure 9: Efficiency as a proportion of maximum performance (higher is better).

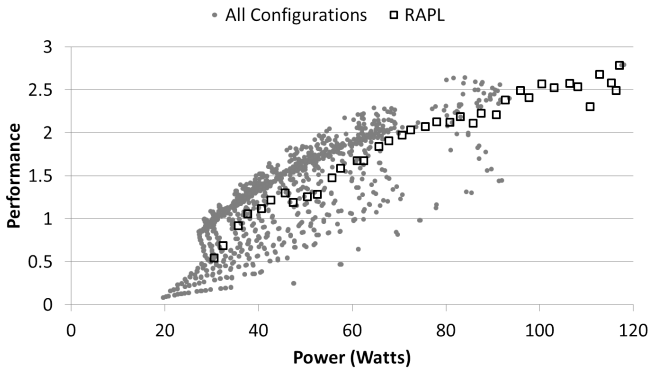

Figure 10: Pareto-Efficiency for HOP.

over hyperthreads.

Fig. 3 illustrates an application for which RAPL performs well across all power limits (jacobi). Fig. 10 shows the same power/performance tradeoff curve for the HOP application. This is an application for which RAPL achieves low efficiency across all limits. The difference in the two figures is apparent. For almost all the black boxes representing the power and performance of RAPL, there are gray dots above those boxes. The presence of those gray dots indicates that higher performance is achievable for the same power consumption.

5. RELATED WORK

Multicore scalability is increasingly limited by power and thermal management [10, 37]. These limitations have inspired a number of techniques for managing power.

Some systems provide performance guarantees while minimizing power consumption. Examples exist at the cluster level [19, 38], and the node level. At the node level, proposed techniques provide performance and minimize power by managing DVFS in the processor [41], core throttling [42], assignment of cores to an application [26], caches [1], DRAM [43], and disks [23]. Researchers have shown that additional power savings can be achieved by coordinating multiple components within a node [27]. For example, Li et al. propose a method for managing memory and processor [24], while Dubach et al. coordinate a large collection of microarchitectural features [9]. Maggio et al. develop a control system for managing core allocation and clock speed [25]. Bitirgen et al. coordinate clock speed, cache, and memory bandwidth [3]. Sharifi et al. develop an adaptive (but non-general) control scheme (called METE) for managing cores, caches and off-chip bandwidth [35]. Several approaches coordinate general sets of resources [18, 20, 28]. All these listed techniques provide performance guarantees (*e.g.,* for meeting quality-of-service or real-time requirements) and minimize power consumption. None of these techniques, however, can guarantee power consumption or meet power budgets.

Some systems guarantee power consumption while maximizing performance subject to the power constraint. Cluster level solutions which guarantee power consumption include those proposed by Wang et al. [39] and Raghavendra et al. [30]. These cluster-level solutions require some node-level power management scheme. Node-level systems for guaranteeing power consumption have been developed to manage different individual components including DVFS for a processor [22], per-core DVFS in a multicore [21], processor idle-time [12, 42], DRAM [8].

It has been noted that coordinated allocation of multiple components should be more efficient than management of components in isolation [14, 15, 17, 27]. Thus approaches have been proposed which provide power guarantees while increasing performance through coordinated management of multiple components, including processor and DRAM [5, 11], processors speed and core allocation [6, 32], and combining DVFS and thread scheduling [31, 40]. Despite differences in mechanisms, these techniques all solve a common problem: select the best configuration for meeting a given power limit. Given these studies, it is not surprising that RAPL would not achieve high efficiency for some applications. We suggest that the methodology described in this paper provides a general framework for evaluating and comparing different power control systems in terms of both their temporal behavior (SASO properties) and delivered performance (efficiency).

We note several other studies have begun to evaluate the RAPL power management system. Rountree et al. explore RAPL as a replacement for DVFS in high-performance computing systems [33]. Sarood et al. use RAPL to set power bounds on across an over-provisioned cluster running homogeneous application processes [34]. Venkatesh et al. use RAPL to measure (but not control) energy consumption in large message-passing applications [36]. While all these studies measure RAPL, none attempts the systematic evaluation of the behavioral measures explored in this paper. For example, Rountree et al. evaluate RAPL's accuracy, but not its stability, settling time, or overshoot. In addition, their efficiency evaluation only compares to DVFS and they do not evaluate other available configurations (such as core and memory usage) affecting performance and power consumption tradeoffs.

6. CONCLUSION & FUTURE WORK

We have evaluated Intel's Running Average Power Limit (RAPL) interface as a control system. We have examined its performance in the four SASO properties desirable for many control systems: stability, accuracy, settling time, and maximum overshoot. In addition, we explored RAPL's efficiency, or ability to deliver performance within a power limit. We discuss the implications of our findings for each of these properties below.

RAPL achieves good **stability** for most long running applications. Stability can be negatively affected by high overshoots for short-lived applications operating at low power limits. In addition, RAPL will not stabilize inherently unstable applications, which is not surprising but important for users to understand.

Accuracy means the power limits are respected, so it is likely the most important attribute for RAPL users. Therefore, it is good to know that RAPL achieves high accuracy in general. The only applications for which RAPL does not achieve high accuracy are those which are inherently unstable. Although it is unrealistic to expect RAPL to stabilize behavior for these applications, it would

be nice if their instability did not negatively impact accuracy.

RAPL achieves low **settling times** for our test applications, relative to their overall runtime. The measured settling times indicate that RAPL may not be suitable to manage power in a system running many short-lived jobs.

RAPL's **maximum overshoot** can be high, especially for small power limits. These overshoots tend to be of short duration. However, even such short durations could be an issue in a large-scale distributed system if all nodes simultaneously overshoot their limits by significant amounts.

Users should know RAPL's **efficiency** depends on both the application running and the power limit. First, RAPL is inefficient at very low power limits (reaching approximately 65% of optimal performance for the 30 Watt budget). Second, RAPL is inefficient for some applications even at higher limits. These applications fall into a class that tend to make inefficient use of one of the other on-chip resources (*e.g.,* , hyperthreading, cores). For such applications, it is possible to achieve higher performance within the power limit by decreasing the resource they use inefficiently and increasing clock speed.

Our work has evaluated RAPL on a single processor with sixteen benchmarks. In future work, we hope to expand this study to multiple machines of different generations, and to explore additional benchmarks. Finally, we hope to compare RAPL to software power control systems, especially those (discussed in Section 5) which are designed to provide greater performance within a power limit by coordinating multiple on chip resources.

Acknowledgements

This work was performed under the Argo project sponsored by the U.S. Department of Energy (contract DE-AC02-06CH11357).

References

[1] R. Balasubramonian et al. “Memory hierarchy reconfiguration for energy and performance in general-purpose processor architectures”. In: *MICRO*. 2000.

[2] C. Bienia et al. “The PARSEC Benchmark Suite: Characterization and Architectural Implications”. In: *PACT*. 2008.

[3] R. Bitirgen et al. “Coordinated management of multiple interacting resources in chip multiprocessors: A machine learning approach”. In: *MICRO*. 2008.

[4] S. Che et al. “Rodinia: A Benchmark Suite for Heterogeneous Computing”. In: *IISWC*. 2009.

[5] J. Chen and L. K. John. “Predictive coordination of multiple on-chip resources for chip multiprocessors”. In: *ICS*. 2011.

[6] R. Cochran et al. “Pack & Cap: adaptive DVFS and thread packing under power caps”. In: *MICRO*. 2011.

[7] H. David et al. “RAPL: Memory Power Estimation and Capping”. In: *ISLPED*. 2010.

[8] B. Diniz et al. “Limiting the power consumption of main memory”. In: *ISCA*. 2007.

[9] C. Dubach et al. “A Predictive Model for Dynamic Microarchitectural Adaptivity Control”. In: *MICRO*. 2010.

[10] H. Esmaeilzadeh et al. “Dark silicon and the end of multicore scaling”. In: *ISCA*. 2011.

[11] W. Felter et al. “A performance-conserving approach for reducing peak power consumption in server systems”. In: *ICS*. 2005.

[12] A. Gandhi et al. “Power capping via forced idleness”. In: *Workshop on Energy-Efficient Design*. Austin, TX, 2009.

[13] J. L. Hellerstein et al. *Feedback Control of Computing Systems*. John Wiley & Sons, 2004.

[14] U. Hoelzle and L. A. Barroso. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. 1st. Morgan and Claypool Publishers, 2009.

[15] H. Hoffmann and M. Maggio. “PCP: A Generalized Approach to Optimizing Performance Under Power Constraints through Resource Management”. In: *ICAC*. 2014.

[16] H. Hoffmann et al. “Dynamic Knobs for Responsive Power-Aware Computing”. In: *ASPLOS*. 2011.

[17] H. Hoffmann et al. “Self-aware computing in the Angstrom processor”. In: *DAC*. 2012.

[18] H. Hoffmann et al. “A Generalized Software Framework for Accurate and Efficient Managment of Performance Goals”. In: *EMSOFT*. 2013.

[19] T. Horvath et al. “Dynamic Voltage Scaling in Multitier Web Servers with End-to-End Delay Control”. In: *Computers, IEEE Transactions on* 56.4 (2007).

[20] C. Imes et al. “POET: A Portable Approach to Minimizing Energy Under Soft Real-time Constraints”. In: *RTAS*. 2015.

[21] C. Isci et al. “An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget”. In: *MICRO*. 2006.

[22] C. Lefurgy et al. “Power capping: a prelude to power shifting”. In: *Cluster Computing* 11.2 (2008).

[23] X. Li et al. “Performance directed energy management for main memory and disks”. In: *Trans. Storage* 1.3 (2005).

[24] X. Li et al. “Cross-component energy management: Joint adaptation of processor and memory”. In: *ACM Trans. Archit. Code Optim.* 4.3 (2007).

[25] M. Maggio et al. “Power optimization in embedded systems via feedback control of resource allocation”. In: *IEEE Transactions on Control Systems Technology (to appear)* ().

[26] M. Maggio et al. “Controlling software applications via resource allocation within the Heartbeats framework”. In: *CDC*. 2010.

[27] D. Meisner et al. “Power management of online data-intensive services”. In: *ISCA* (2011).

[28] N. Mishra et al. “A Probabilistic Graphical Model-based Approach for Minimizing Energy Under Performance Constraints”. In: *ASPLOS*. 2015.

[29] R. Narayanan et al. “MineBench: A Benchmark Suite for Data Mining Workloads”. In: *IISWC*. 2006.

[30] R. Raghavendra et al. “No "power" struggles: coordinated multi-level power management for the data center”. In: *ASPLOS*. 2008.

[31] K. K. Rangan et al. “Thread motion: fine-grained power management for multi-core systems”. In: *ISCA*. 2009.

[32] S. Reda et al. “Adaptive Power Capping for Servers with Multithreaded Workloads”. In: *Micro, IEEE* 32.5 (2012).

[33] B. Rountree et al. “Beyond DVFS: A First Look at Performance under a Hardware-Enforced Power Bound”. In: *IPDPSW*. 2012.

[34] O. Sarood et al. “Optimizing power allocation to CPU and memory subsystems in overprovisioned HPC systems”. In: *CLUSTER*. 2013.

[35] A. Sharifi et al. “METE: meeting end-to-end QoS in multicores through system-wide resource management”. In: *SIGMETRICS*. 2011.

[36] A. Venkatesh et al. “Evaluation of Energy Characteristics of MPI Communication Primitives with RAPL”. In: *IPDPSW*. 2013.

[37] G. Venkatesh et al. “Conservation cores: reducing the energy of mature computations”. In: *ASPLOS*. 2010.

[38] A. Verma et al. “Server workload analysis for power minimization using consolidation”. In: *USENIX Annual technical conference*. 2009.

[39] X. Wang et al. “MIMO Power Control for High-Density Servers in an Enclosure”. In: *IEEE Transactions on Parallel and Distributed Systems* 21.10 (2010).

[40] J. A. Winter et al. “Scalable thread scheduling and global power management for heterogeneous many-core architectures”. In: *PACT*. 2010.

[41] Q. Wu et al. “Formal online methods for voltage/frequency control in multiple clock domain microprocessors”. In: *ASPLOS*. 2004.

[42] X. Zhang et al. “A Flexible Framework for Throttling-Enabled Multicore Management (TEMM)”. In: *ICPP*. 2012.

[43] H. Zheng et al. “Mini-rank: Adaptive DRAM architecture for improving memory power efficiency”. In: *MICRO*. 2008.