# Power and Energy Containers for Multicore Servers

Kai Shen
University of Rochester
kshen@cs.rochester.edu

Arrvindh Shriraman
Simon Fraser University
ashriram@cs.sfu.ca

Sandhya Dwarkadas
University of Rochester
sandhya@cs.rochester.edu

Xiao Zhang*
University of Rochester
xiao@cs.rochester.edu

## ABSTRACT

Power capping and energy efficiency are critical concerns in server systems, particularly when serving dynamic workloads on resource-sharing multicores. We present a new operating system facility (*power and energy containers*) that accounts for and controls the power/energy usage of individual fine-grained server requests. This facility is enabled by novel techniques for multicore power attribution to concurrent tasks, measurement/modeling alignment to enhance predictability, and request power accounting and control.

## Categories and Subject Descriptors

C.5.5 [**Computer System Implementation**]: Servers; D.4.8 [**Operating Systems**]: Performance—*Measurements, Modeling and prediction*

## Keywords

Multicore, server system, power and energy management.

## 1. INTRODUCTION

Online applications continuously evolve with new features and some (like social networks, Wiki sites, and online collaboration) rely on end users to supply content or even content-generating code. The workload diversity and dynamic client-directed processing result in large power fluctuations on modern hardware with increasing specialization and power proportionality. Recognizing the resource usage of individual requests facilitates fine-grained attribution of energy usage to clients, helping expose the full cost of web usage. Furthermore, high power-consuming tasks (or "power viruses" [2]) may either appear accidentally or be maliciously devised. Isolating per-client power attribution to identify such tasks so as to cap the system power draw in a fair fashion is desirable.

Building on our past work [3, 5] on per-request behavior characterization in a multicore server, we address the additional challenges of per-request power/energy accounting and management. Request executions in a concurrent, multi-stage server contain fine-grained activities with frequent context switches. Direct power measurements on such spatial and temporal granularities are not available on today's systems. Concurrent task executions, dynamic workload behaviors, and multicore hardware sharing lead to additional complexities in modeling per-task power behavior. Request-level power and energy management also requires agile, low-cost control to ensure isolation and achieve efficiency.

*Zhang is currently affiliated with Google.


*SIGMETRICS'12,* June 11–15, 2012, London, England, UK.
ACM 978-1-4503-1097-0/12/06.

We address these challenges in a new operating system facility, called *power and energy containers*, which both tracks and allows control of the power/energy usage of individual requests in multicore servers. Our power and energy containers enable fine-grained, agile management of multicore server power/energy resources in unprecedented ways. Below, we present design highlights and a brief case study, with more details in a technical report [4].

## 2. DESIGN HIGHLIGHTS

We support power attribution and control in two new dimensions—1) over concurrent executions on a shared-resource multicore and 2) among fine-grained requests in a multi-stage server application.

### *Power Attribution to Concurrent Tasks.*

Bellosa [1] validated a linear model between the system power consumption and the frequency of hardware events such as instructions per cycle and memory transaction rate. This suggests that the power accounting for a task can linearly correlate with physical event rates at the CPU core that the task runs on. On a multicore processor chip with intricately shared hardware resources, however, chip-wide environmental factors also affect per-core power accounting. In particular, the maintenance of shared multicore resources (including at least clocking circuitry and voltage regulators) consumes some active power as long as one core is running. However, the power consumption does not change proportionally with core-level event rates. Our measurements show that the power increment from idle to one utilized core is much higher than further power increments, which suggests an active power component that does not scale proportionally with core-level physical events.

It is intuitive to evenly attribute the chip maintenance power at each time instant to the currently running tasks. The system utilization level fluctuates over time in production server environments. Proper accounting and attribution of shared chip maintenance power is challenging because one task's share may change depending on activities (or the lack thereof) on other cores. If a core is busy while all other siblings are idle, the full chip maintenance power should be attributed to the task on the busy core. If multiple ($k$) cores are busy at a moment, then each running task on one of the busy cores has $\frac{1.0}{k}$ share of the maintenance power. Note that the chip maintenance power share does not correspond to any processor hardware counter and depends instead on time-synchronized busy states across all CPU cores. The operating system must monitor all CPU busy states to accurately account for this.

### *Measurement Alignment for Model Recalibration.*

Despite the wide uses of event-based power models [1], we found that large errors may arise in practice. Beyond superficial problems like insufficient coverage of modeled events, modeling inaccuracy also results from differing characteristics between calibration and

production workloads. This is particularly the case for unusually high power-consuming production workloads that demand careful attention in server system power management. To address the modeling inaccuracy, we utilize online power measurements to adjust and recalibrate the offline-constructed power models.

The challenge is that while whole-system power measurements can be acquired through off-the-shelf meters, measurement results often arrive after some meter reporting delay and data I/O latency. On the other hand, processor event counter-based power models can be maintained at the much shorter latency of reading CPU registers and computing simple statistics. The measurement results and modeling estimates need to be properly aligned in order to identify modeling errors and recalibrate the model. Our work finds that such alignment can be enabled using signal processing cross-correlation and that the aligned measurement can effect model recalibration and improve modeling accuracy.

*Request Power Accounting and Control.*

We construct OS mechanisms to support request-level power/energy containers in a server system. We track request execution contexts so that power-relevant metrics can be properly attributed and control mechanisms can be properly applied. We identify a request context on-the-fly with recognized patterns on how it propagates over multiple server stages. Specifically, we recognize request context propagation events as those that indicate causal dependences through data and control flows between processes, including socket calls, IPCs, and task forking. By sampling events at propagation boundaries, our power model estimates request power consumption based on the collected event metrics. The cumulative energy usage can be simply calculated as the integral of power over time.

In addition to per-request event sampling, the on-the-fly request tracking allows us to selectively apply power and energy control mechanisms on certain requests. For example, we can apply a particular CPU duty-cycle modulation level to a given request execution. Effectively, when the request starts executing on a CPU core, the core duty-cycle level will be set appropriately. When the CPU core switches to run another request, the core duty-cycle level will be adjusted according to the policy setting of the new request. Our mechanism can limit the power consumption of selected requests without hurting the performance of others. We can similarly support request-specific dynamic voltage/frequency scaling if the multicore processor allows such scaling on a per-core basis.

## 3. POWER VIRUS MANAGEMENT

Our power containers enable fair power conditioning on a multicore server. While CPU duty-cycle modulation can control surging power consumption, indiscriminate full-machine throttling would lead to slowdowns of all running requests regardless of their power use. Without per-request power containers, the occurrence of a power virus could force speed reductions on all concurrently running normal requests. In a power container-controlled system, we maintain a power consumption target for each request. Those that exceed the specified target will be subject to request-specific CPU duty-cycle modulation while other requests will run at full speed. Our experiment runs a server on a quad-core Intel Nehalem machine. The workload fully utilizes all four cores on the machine. In the middle of the experimentation, we inject some high-power requests to mimic power viruses. Figure 1(A) shows that the introduction of power viruses lead to substantial power spikes. We apply our container-based fair power conditioning with a system active power target of 40 Watts. Figure 1(B) shows that our request container-enabled power conditioning can effectively keep power consumption at or below the target level despite the power viruses.

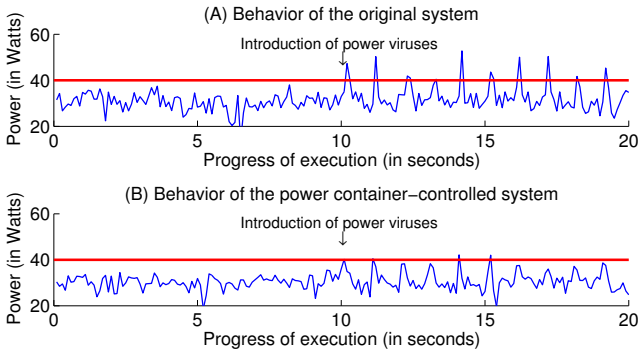


**Figure 1: Measured active (full minus idle) power of a quad-core Nehalem machine. Results are for original (A) and power container-controlled (B) executions with power viruses.**

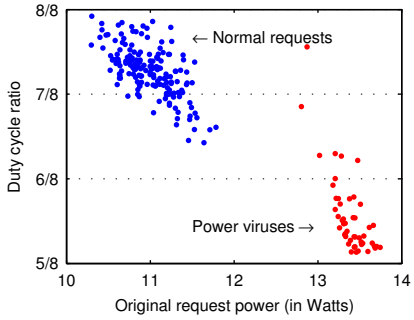


**Figure 2: Each point represents a sample request (normal or power virus). X-coordinate indicates the original (before throttling) request power consumption. Y-coordinate indicates the CPU duty-cycle (slowdown) ratio applied to the request.**

We next show that the CPU speed adjustment has been applied fairly to each request. Figure 2 plots the applied CPU duty-cycle ratio and original request power (before throttling) for each request. Since a request power consumption may fluctuate over its execution, different duty-cycle levels may be applied over time. We show the time-averaged duty-cycle ratio for each request. We also estimate its power consumption in the original system. Results show that the normal server requests suffer small CPU speed slowdown (averaged at about 9%). At the same time, the power viruses are subject to more substantial (31% on average) slowdown.

## 4. ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation (NSF) grants CCF-0448413, CNS-0834451, CCF-0937571, and CCF-1016902. Shen was also supported by an IBM Faculty Award and a Google Research Award.

## 5. REFERENCES

[1] F. Bellosa. The benefits of event-driven energy accounting in power-sensitive systems. In *SIGOPS European Workshop*, 2000.

[2] K. Ganesan, J. Jo, W. L. Bircher, D. Kaseridis, Z. Yu, and L. K. John. System-level max power (SYMPO): a systematic approach for escalating system-level power consumption using synthetic benchmarks. In *PACT*, 2010.

[3] K. Shen. Request behavior variations. In *ASPLOS*, 2010.

[4] K. Shen, A. Shriraman, S. Dwarkadas, and X. Zhang. Power and energy containers for multicore servers. Technical Report #970, Dept. of Computer Science, Univ. of Rochester, 2011.

[5] K. Shen, M. Zhong, S. Dwarkadas, C. Li, C. Stewart, and X. Zhang. Hardware counter driven on-the-fly request signatures. In *ASPLOS*, 2008.