Star-Cap: Cluster Power Management Using Software-Only Models

John D. Davis[1], Suzanne Rivoire[2], and Moises Goldszmidt[1]
Microsoft Research - Silicon Valley Lab[1], Sonoma State University[2]

Abstract

Star-Cap is a high-fidelity, real-time power management system for computer clusters. Star-Cap's cluster power management module uses software-only power models to implement a proactive power capping mechanism with the ability to distribute a cluster's power budget non-uniformly across nodes. We evaluated Star-Cap on a variety of MapReduce-style workloads running on low-power mobile and server systems. Depending on application and platform, Star-Cap improves cluster throughput by 14-43% compared to using a uniform power cap distribution policy with a purely reactive power capping mechanism. Furthermore, Star-Cap maintains cluster throughput while reducing overall cluster power budget by 14% with no additional capital or operating cost. Star-Cap's proactive power capping mechanism also improves system response time to power cap violations by a factor of 2 to 10. Star-Cap's overhead for collecting, computing, and reporting data is less than 1% CPU utilization.

1. Introduction

Current data centers cost about US$10 million per megawatt to build, and around 80% of the facility cost is related to power infrastructure [15]. Active server power management can enable high utilization of this expensive power infrastructure and at the same time avoid the possibility of a catastrophic failure. Unfortunately, current commercial server power management solutions involve additional hardware and software [16, 26], which can double the cost of a server over its 3-5 year lifespan. For example, the low-end power capping solution from HP, which limits power cap violations to periods of 30 seconds or less, costs several hundred US dollars per year per server and can only apply the power cap uniformly across machines in a rack [16].

In this work, we present a power capping framework, Star-Cap, that incorporates software-based models of power consumption, rather than physical measurements, to enforce system-level power budgets. By using high-fidelity, low-overhead models, Star-Cap removes the need for physical measurement infrastructure to implement power capping. The ability to cap power based on models alone is particularly useful for new low-cost server designs, like the Open Compute Project, that have no means to monitor their own power consumption [25].

Star-Cap also allows the power caps for individual machines to adapt to the demands of the workload. Furthermore, large-scale applications generally map different functions to machines within the same rack for locality, which can be exploited by this non-uniform allocation policy. The results we have obtained in applying Star-Cap to a variety of platforms and applications are:

1) For a given power budget, a non-uniform capping policy improves cluster throughput by 14-43% compared to a reactive, uniform capping policy.
2) Star-Cap reduces violations by a factor of 2-10 compared to a reactive, uniform power capping policy using power meters.
3) The overhead of the power modeling and management is less than 1% CPU utilization.

Given these results, we propose the Star-Cap techniques as a worthy alternative to measurement-based techniques. Star-Cap can improve efficiency while reducing data center power infrastructure costs.

The rest of this paper is organized as follows. We address related work in Section 2. Section 3 describes our clusters, workloads, measurement infrastructure and modeling technique. Section 4 presents our software system used for power capping, and Section 5 evaluates this system. We conclude with Section 6.

2. Background

Fan et al. estimated that capping power 1-2 percent of the time would allow them to fit 11 to 24 percent more servers in a fixed power budget [9], but they did not propose specific mechanisms for doing so. Power cap-

Table 1. 5-machine cluster details. *Maximum DRAM capacity.

	Mobile Cluster	Server Cluster
Processor	Intel Core 2 Duo, 2-core, 2.26 GHz	AMD Opteron, dual, 4-core, 2.0 GHz
Power Range	25-46 W (TDP: 25W)	135-190 W (TDP 50W)
DRAM	4 GB DDR3-1066*	32 GB DDR2-800
Disk	1 Micron SSD	2 10K RPM SATA
OS &FS	Win2K8 Server R2, NTFS	

ping with Star-Cap is a mechanism to realize their goal of fully utilizing the data center power infrastructure.

Of the approaches to power capping that have been proposed, ours is most similar to the ad-hoc preemptive and reactive policies presented by Ranganathan et al. [29]. Because our power models use more features that contribute to dynamic full-system power, resulting in a more accurate model, we can achieve better performance and less oscillation for high-utilization workloads. Alternatively, several researchers have proposed control-theoretic approaches to power capping [31, 20, 21, 10, 27]. These approaches may provide better guarantees of performance and stability, but at the price of increased complexity.

Previous power capping studies have used either measured power [20] or simple models based on CPU frequency and/or utilization [29, 12, 22, 10, 3]. Most previous work has used linear models [9, 4, 19] or models that do not capture interaction between predictors [3]. Our work is the first to apply more comprehensive high-level models to capture the systems' full dynamic range and account for system variability [6, 7, 28].

Most previous work in power capping has used processor frequency state as the main or only actuator [31, 12, 29, 20, 10, 3, 13, 27, 30] because DVFS is widely available and because the processor is the main dynamic power consumer. We also use DVFS to demonstrate Star-Cap capabilities, yet our models are based on a variety of system metrics and not solely the CPU. Nothing prevents Star-Cap from being extended to exploit other power management hooks when available. Promising candidates include core parking, component-level low-power states [8], application-level throttling [17], low-power hard drive modes [14], and low-power networking modes [1].

3. Hardware and software infrastructure

We trained power models and deployed the Star-Cap system on the five-node homogeneous clusters described in Table 1. We only present results in Section 5 for the mobile cluster because the results for the server cluster are similar. We ran an assortment of workloads using the Dryad and DryadLINQ distributed application framework [18, 32]. Some workloads are CPU-intensive, while others are dominated by disk or network activity. To build the power models, we used a tenth of the data for training and the rest for testing. The workloads used are:

- **Sort.** This workload sorts 4GB of data with 100-byte records. This workload has high disk and network utilization.
- **PageRank.** This workload runs a graph-based page ranking algorithm over the ClueWeb09 dataset [5], a corpus of about 1 billion web pages. PageRank has high network utilization.
- **Prime.** This workload checks for primeness of each of approximately 1,000,000 numbers on each of 5 partitions in a cluster. This workload is CPU-intensive and produces little network traffic.
- **WordCount.** This workload reads through 500 MB text files on each of 5 partitions in a cluster and tallies the occurrences of each word that appears. It produces little network traffic.

Hardware: Every machine in every cluster is individually instrumented with a power meter. We use the WattsUp? Pro digital power meter to capture the wall power once per second [23]. Each machine reads its own power measurements over a USB port. The power meters have an error of 1.5%.

Software: Each system runs Windows Server 2008 R2, which has a convenient and standardized OS-level performance counter interface. We use Windows Perfmon to record measurements once per second for Windows ETW (Event Tracing for Windows) software counters as well as the WattsUp? Pro power meter readings.

3.2. Modeling methods

We use a quadratic power model (QPM) with a cluster-specific feature set. These models have been built for a variety of systems and have been shown to be the most accurate high-level models across a variety of clusters [7]. This model provides a balance between low overhead (less than 1% CPU utilization), and high accuracy in the presence of variability [6]. We use Multivariate Adaptive Regression Splines (MARS) algorithm to automatically fit the quadratic model parameters from training data [11].

Quadratic power model:

$$\hat{f}(x_1, \ldots, x_n) = a_0 + \sum_i \sum_j a_{i,j} \times B_i^s(x_i, t_i) \times B_j^s(x_j, t_j)$$

In the quadratic model, the parameter s can be positive (+) or negative (−), and the basis functions $B_{i,j}^{\ s}$ are hinge functions. $B_{i,j}^{+}(x,t)$ takes a value of 0 if $x \leq t$ and a

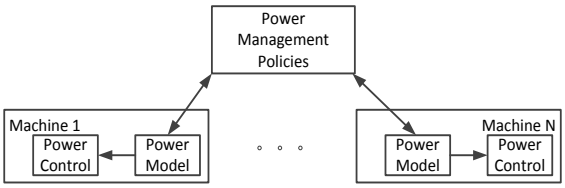

Figure 1. Two-level power management system.

value of $x–t$ otherwise. Similarly, $B_{i,j}^{-}(x,t)$ takes a value of 0 if $x > t$ and $t–x$ otherwise. The t thresholds are called *knots* and the j indices permit a feature to be responsible for multiple knots. Fitting these models requires finding the knots $t_{i,j}$ and the parameters $a_{i,j}$. The MARS algorithm [11] selects the knots and fits the parameters, determining how the basis functions interact. We restrict this interaction to degree = 2.

4. Star-Cap: Software-only cluster power capping

Star-Cap is an extensible two-level power management framework that separates the allocation of power budgets from the specific power capping implementation. In this section, we describe the Star-Cap framework and the control mechanisms and algorithms used to enforce the power budget on individual nodes.

4.1 Framework

The two-level architecture of Star-Cap is shown in Figure 1. The top (cluster) level allocates the cluster's power budget among the individual nodes, providing each machine with a power cap target in Watts. At the top level, we experiment with two policies for allocating the power budget. The first is a uniform policy, in which the power budget is evenly distributed across the nodes in the cluster. The second is a non-uniform policy, which allows different machines in the cluster to have different power caps. This policy adjusts for load imbalances across or within applications, granting higher power caps to busier nodes.

The bottom (machine) level of the Star-Cap framework is responsible for enforcing the power cap target set by the top-level controller. This organization decouples the power management policy from the mechanism, providing flexibility in the power capping implementation. The machine-level power capping software must monitor the machine's current power consumption and adjust its power management knobs as needed to stay within the power budget. In our implementation, we adjust the available processor frequency states by using the Windows Power Management functions [24], since the processors are the main consumers of dynamic power in our system and the easiest components to control in software. However, as high-level power management knobs for other components become available, we envision using our models to determine which knobs to adjust to reach the power management target.

The remainder of this section describes the machine-level power capping policies we implemented. These policies are evaluated in Section 5.

4.2. Control mechanisms

All of the control algorithms we implemented use the same method of adjusting the machine-level power consumption, and they all share three common inputs. The following inputs are common to all of the control algorithm:

- A full-system power cap in Watts (P_{target}), set by the top-level controller
- A measurement or estimate of the current power consumption ($P_{current}$)
- The set of processor frequency states that are currently permitted ($\{f_0 \dots f_k\}$), as described below

In all of our implementations of Star-Cap, the power capping software adjusts the power consumption by restricting or increasing the range of available processor frequency states. If the processor supports a set of frequency states $\{f_0 \dots f_n\}$, then the power capping software will restrict the OS to frequency states f_0 through f_k, where $k \leq n$. In our implementation, we allow the OS's existing on-demand power management policy to choose the exact frequency state from the subset of states allowed by the power capping software. Our current implementation uses the same set of permitted frequency states for all cores in the system.

4.3. Control algorithms

Our control algorithms require two configurable power consumption thresholds, P_{hi} and P_{lo}. We configure P_{hi} to be 95% of the power cap and P_{lo} to be 90% of the power cap. Our algorithms make the following adjustments:

- If $P_{current} > P_{hi}$, adjust the available frequency states from $\{f_0 \dots f_k\}$ to $\{f_0 \dots f_{k-1}\}$.
- If $P_{current} < P_{lo}$, adjust the available frequency states from $\{f_0 \dots f_k\}$ to $\{f_0 \dots f_{k+1}\}$.

Thus, we adjust the number of allowed frequency states by one step at a time over all cores in the system.

We implement two variations of this algorithm: (1) ReCap (Reactive Power Capping), a window-based control technique that reacts to $P_{current}$, and (2) ProCap (Proactive Predictive Power Capping), which reacts to $P_{current}$ and one additional input. This additional input, P_{future}, is the power consumption predicted by the quadratic power model after the change in available frequency states.

ReCap details. ReCap checks $P_{current}$ once per second to determine whether or not to adjust the frequency state. After adjusting the frequency states, it waits for a period of time for the system power to settle; we empirically found that a window of 10 seconds was necessary to observe the change of frequency state's effects on the full-system power consumption.

We experiment with three different versions of ReCap using our three different methods of determining $P_{current}$:

- M-ReCap: $P_{current}$ is the actual measured full-system power consumption.
- L-ReCap: $P_{current}$ is the estimated power using a simple linear model based only on CPU utilization.
- C-Recap: $P_{current}$ is the estimated power using the cluster-specific quadratic power model described in Section 3.2.

ProCap details. Like ReCap, ProCap checks $P_{current}$ once per second. However, before adjusting the available frequency states, it will use the quadratic power model to predict P_{future}. This predicted power consumption is based on the processor frequency state changing to f_{k-1} or f_{k+1} and all other resource utilizations remaining constant. If the predicted P_{future} is outside the range $P_{lo} \leq P_{future} \leq P_{hi}$, then ProCap will not adjust the range of available frequency states. This simple technique helps to prevent oscillations in power consumption and removes the need to wait for the power consumption to settle to its new value. Thus, ProCap does not use a history window.

The next section evaluates these specific techniques. More complicated prediction and control mechanisms are possible and can be layered on top of our power models. However, the simple techniques implemented here shed light on the potential of model-based power capping schemes that do not rely on physical measurement.

5. Evaluation

We report results on the Intel Core 2 Duo cluster, as it has a relatively large dynamic power range and similar per-core power to future low-power server platforms. Note that using QPMs for the server cluster yielded similar results, which have been excluded for brevity. The general nature of the QPMs makes our power capping system generally applicable. We could extend power capping to any platform with QPMs and in particular all the platforms studied in [7].

Our dual-core processor has four frequency states: 1596 MHz (70%), 1862 MHz (82%), 2128 MHz (94%), and 2261 MHz (100%). Figure 2 shows the measured power profiles of each machine for the four workloads in this study when the power is capped at each of the four frequency states. Altogether, the power consumption ranges from 36 to 44 W per machine, or 180 to 220 W for the cluster. Figure 2 also shows the machine-to-machine full-system power variation for these workloads. Overall, Star-Cap uses about 1% CPU utilization on the mobile platform.

5.1. Power capping profiles

The following figures illustrate our results (a) that accurate models are required for fine-grained management of system/cluster power, and (b) that proactive power capping using our models is much better than reactive power capping, even when reactive capping is based on direct power measurements. For clarity, we graph the measured power of a single node, since all the nodes have similar characteristics. The solid lines are the result of using measured power for power capping and the lines with markers are the measured power overlaid when using ReCap and ProCap for power capping. The horizontal dotted line on each graph represents the machine power cap.

Low power cap results: Figure 3 shows the results of capping power with a value of 38 W per node for the four workloads. Figure 3 (A) shows reactive power capping techniques based on the actual measured power (M-ReCap) and predicted cluster power (C-ReCap). This comparison shows that the QPM is accurate enough to replace direct measurement, thereby lowering server cost. Figure 3 (B) shows M-ReCap compared to the proactive power capping algorithm (ProCap). ProCap has fewer power cap violations, and its violations have shorter duration.

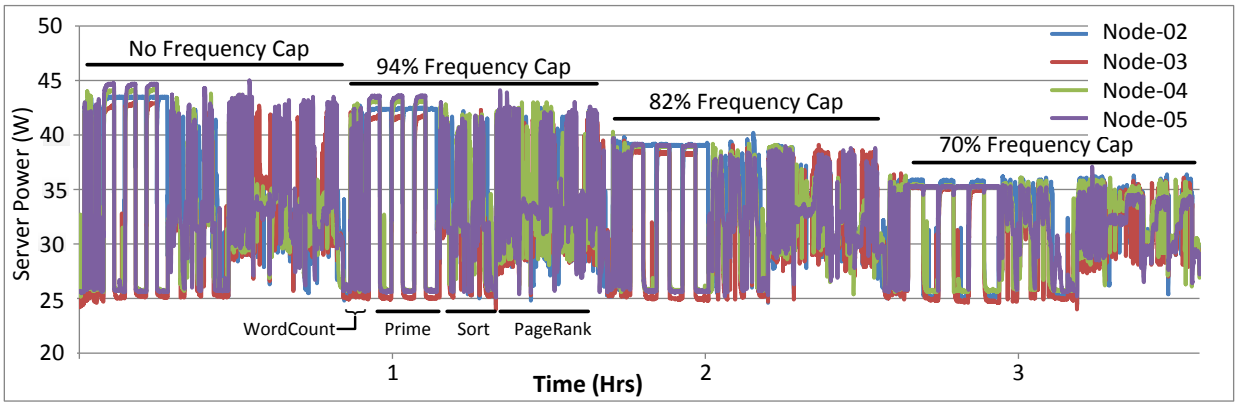

Figure 2. Workload power characteristics for different values of the maximum processor frequency, from left to right.

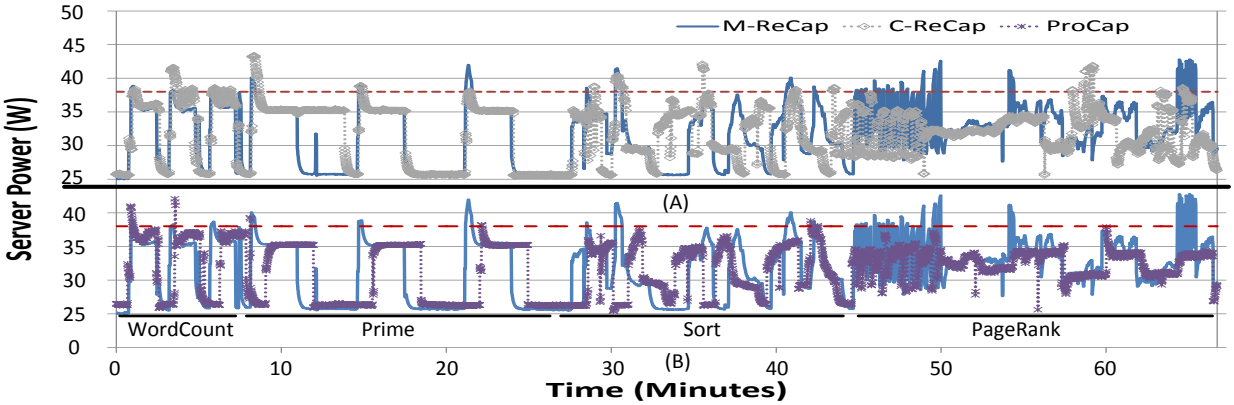

Figure 3. 38W per-node power cap enforced using (A) reactive capping using measured power (M-ReCap) vs. cluster-specific power model (C-ReCap), and (B) M-ReCap vs. proactive capping a using cluster-specific power model (ProCap).

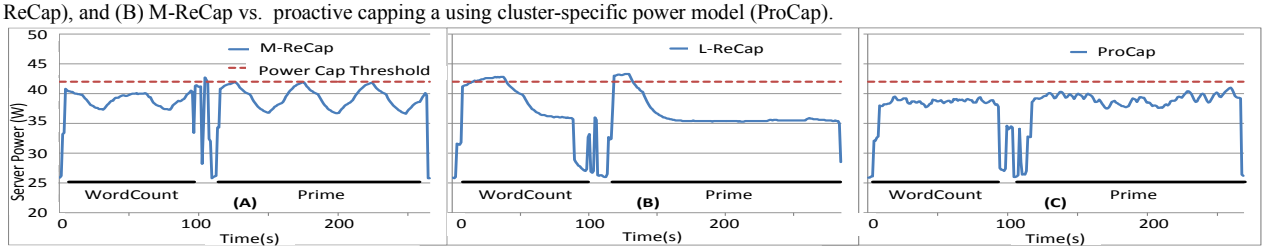

Figure 4. 42W power cap examples for WordCount and Primes using the Reactive Power Capping with (A) (M-ReCap) measured power and (B) (L-ReCap) a CPU utilization linear model, showing its bimodal operation, and (C) the ProCap cluster-specific machine power model predicting the possible future power states.

Figure 3 (A) contains large overshoot peaks that coincide with the duration of the history window. Furthermore, even with a large history window compared to the sampling rate, both reactive algorithms still experience some oscillations. This is because, for some power cap values, there is no frequency state that reliably leads to power consumption between the low and high thresholds, so the controller oscillates between the two states. PageRank exhibits this behavior and is the most violation-prone workload (right side of Figure 3).

For applications that exhibit clear phase behavior, the proactive algorithm (ProCap) is superior to the reactive (C-ReCap) algorithm. In this scenario, the current state of the system is a very good proxy for the next state of the system at time t+1. This scheme still experiences overshoot peaks, but they are smaller both in magnitude and duration by factors of 2-10. Furthermore, ProCap does not experience the same type of oscillations at high CPU utilization that the reactive schemes do. ProCap is able to lock on and hold a particular power state instead of constantly reacting and correcting.

High power cap results: Figure 4 shows the behavior when we increase the power cap threshold to 42W. It zooms in to the WordCount and Prime workloads since they have shorter runtimes, and their behavior is representative of the complete set of workloads. Figure 4 (A) again shows oscillations when using the M-ReCap algorithm. We omit C-Recap because it once again shows almost identical behavior to M-Recap.

In Figure 4 (B), we show reactive power capping based on a strawman: a simple linear model based on CPU utilization alone (L-Recap). This model has the characteristic overshoot of the other reactive algorithms. Moreover, on closer examination, it is clear that L-ReCap forces the processor into the lowest frequency possible, operating around 35W. The reason is that, without CPU frequency information, L-ReCap overpredicts power based on the high CPU utilization. This demonstrates the bimodal power capping behavior of low fidelity models. L-ReCap is unable to take advantage of intermediate power management modes in these scenarios. For example, setting the machine power cap value above 42W for the L-ReCap algorithm resulted in no power capping. This is because the range of the linear model does not match the dynamic range of the machine, so the model is unable to accurately predict the system power at high frequency and utilization.

5.2. Power cap violations

The cluster-level power cap violations are generally at least 50% lower than the per-machine violations, demonstrating that the machine violations are not coincident for these workloads even though the tasks run on these machines are coordinated. The Prime, WordCount, and Sort workloads are well suited for the ProCap mechanism using the QPM. We also compared these algorithms to a static maximum processor frequency limit, derived empirically. To not violate a 38 W or 40 W power cap, the maximum operating frequency must be set to 1596 MHz (the lowest operating frequency), and at 42 W, the maximum operating frequency cannot exceed 1862 MHz. In general, the ProCap algorithm has the lowest violation percentage of the dynamic power capping mechanisms for our workloads. Overall, the power capping mechanisms provide viable alternatives to static frequency capping that are easier to implement without any *a priori* knowledge of the system. Furthermore, the ProCap mechanism combined with non-uniform power capping, as described in the next section, reduces application runtime compared to statically capping CPU frequency.

There are scenarios where ProCap exhibits some power cap violations. For applications with sporadic behavior, like PageRank, the current state of the system is not a good indicator of the future state, and as such, power predictions will err. To compensate, we propose a *Hybrid Proactive Predictive Power Capping* (ProCap+) algorithm, where we add a violation window that decrements the maximum allowable P-state when the number of violations occurring within the window exceeds a configurable threshold.

Table 2. Normalized runtime of the workloads with no power cap, fixed processor frequency, a uniform power cap (190 W), and a non-uniform power cap (190 W). PR: PageRank, WC:WordCount.

	PR	Prime	Sort	WC
No Power Cap (220W)	1	1	1	1
MHz Cap: 1596 MHz	1.34	1.43	1.22	1.25
Uniform Power Cap	1.20	1.42	1.24	1.27
Non-uniform Power Cap	1.08	1	1.08	1.04

5.3. Power capping and performance

Table 2 provides the normalized performance of various scenarios compared to running the workloads without a power cap (row 1). When running workloads at the slowest processor frequency, we observe an increase in runtime by 34%, 43%, 22% and 25%, for PageRank, Primes, Sort, and WordCount, respectively (row 2). These numbers are consistent with the workload characteristics; CPU-bound applications like Prime suffer more than I/O- and network-bound workloads.

The next two rows of Table 2 show different options for distributing the cluster power cap on a workload with some load imbalance. Row 3 shows the runtime when applying the power cap uniformly across machines, for a per-machine target of 38W. This low cap forces the machines to operate mostly in the lowest P-state.

Row 4 shows the results of distributing the cluster power budget non-uniformly across nodes. In this example, nodes with more work to do are given a higher power cap than the others, avoiding most of the runtime slowdown. In the case of our cluster, we can decrease the power cap of the job scheduler node and/or worker node(s) and increase another worker node commensurately, maintaining the same cluster power cap of 190W. The non-uniform power capping regained between 14-43% of overall performance when compared to the uniform power capping scenarios, while maintaining a power budget that is 14% lower than the NoCap power budget. This non-uniform policy translates into lower power infrastructure costs and/or the ability to host significantly more machines [9].

6. Conclusions

Our results show that power capping schemes can be improved by using power models to anticipate the effect of a possible frequency state change. Our proactive power capping mechanism provided 2-10 times better response time than reactive power capping mechanism that use direct power measurement. Significant data

center power spikes can occur at the granularity of a minute [2], and Star-Cap mitigates these spikes with a response time of 20 seconds or less for our workloads. Furthermore, using non-uniform power capping, we are able to improve cluster throughput by 14-43% by borrowing power from other machines and increasing the power cap at the granularity of a single system. To the best of our knowledge, this study is the first to apply high-level, full-system and full-cluster power models to power capping. This allows us to implement power capping with no additional hardware support, dramatically reducing the server cost with minimal overhead (1% of CPU utilization). Although not explicitly evaluated in this work, we believe that by decoupling the power capping policy from the power capping mechanism, our system can also be used to manage heterogeneous clusters and enable rapid deployment of new power capping mechanisms at fine granularity.

7. References

[1] D. Abts et al., "Energy Proportional Datacenter Networks," in *ISCA,* 2010.

[2] L. A. Barroso, "Warehouse-Scale Computing Entering the Teenage Decade," *FCRC* plenary talk*,* 2011.

[3] Y. Chen et al., "Managing server energy and operational costs in hosting centers," in *SIGMETRICS*, 2005.

[4] J. Choi et al., "Profiling, prediction, and capping of power consumption in consolidated environments," in *MASCOTS*, 2008.

[5] ClueWeb09 dataset. Available at http://boston.lti.cs.cmu.edu/Data/clueweb09/

[6] J. D. Davis et al., "Accounting for variability in large-scale cluster power models," in *EXERT*, 2011.

[7] J. D. Davis et at., "CHAOS: Composable Highly Accurate OS-based Power Models," in IISWC 2012.

[8] Q. Deng et al., "MemScale: Active low-power modes for main memory," in *ASPLOS*, 2011.

[9] X. Fan et al, "Power provisioning for a warehouse-sized computer," in *ISCA*, 2007.

[10] M.E. Femal and V.W. Freeh, "Safe overprovisioning: Using power limits to increase aggregate throughput," in *Intl. Wkshp. on Power-Aware Computer Systems*, 2004.

[11] J. Friedman, Multivariate Adaptive Regression Splines, 1991, *The Annals of Statistics*, 19:1-141.

[12] A. Gandhi et al., "Optimal power allocation in server farms," in *SIGMETRICS*, 2009.

[13] S. Govindan et al., "Statistical profiling-based techniques for effective power provisioning in data centers," in *EuroSys*, 2009.

[14] S. Gurumurthi et al., "DRPM: Dynamic Speed Control for Power Management in Server Class Disks," in *ISCA*, 2003.

[15] J. Hamilton, "Annual fully burdened cost of power," Dec. 2008. Available at: http://perspectives.mvdirona.com/2008/12/06/AnnualFullyBurdenedCostOfPower.aspx

[16] Hewlett-Packard, "HP power capping and dynamic power capping for ProLiant servers," HP Technical Report TC090303TB, 2009.

[17] H. Hoffmann et al., "Dynamic knobs for responsive power-aware computing," in *ASPLOS*, 2011.

[18] M. Isard et al., "Dryad: distributed data-parallel programs from sequential building blocks," in *EuroSys*, 2007.

[19] A. Kansal et al., "Virtual machine power monitoring and provisioning," in *Symp. on Cloud Computing*, 2010.

[20] C. Lefurgy et al., "Power capping: A prelude to power shifting," in *Cluster Computing*, 11(2):183-195, 2008.

[21] T. Li and L. K. John, "Run-time modeling and estimation of operating system power consumption," in *SIGMETRICS*, 2003.

[22] H. Lim et al., "Power budgeting for virtualized data centers," in *USENIX ATC,* 2011.

[23] Microsoft, "Joulemeter," available at: http://research.microsoft.com/en-us/downloads/fe9e10c5-5c5b-450c-a674-daf55565f794/

[24] Microsoft, "Power management references," available at: http://msdn.microsoft.com/en-us/library/aa373170(v=VS.85) .aspx

[25] J. Park, "Data Center v1.0," available at: http://opencompute.org/wp/wp-content/uploads/2011/07/Data Center-Electrical-Specifications.pdf, April, 2011

[26] P. K. Popa, "Managing server energy consumption using IBM PowerExecutive," IBM white paper, 2006.

[27] R. Raghavendra et al., "No 'power struggles': Coordinated multi-level power management for the data center," in *ASPLOS*, 2008.

[28] K. Rajamani et al., "Power Management for Computer Systems and Datacenters," in *ISLPED*, 2008.

[29] P. Ranganathan et al., "Ensemble-level power management for dense blade servers," in *ISCA*, 2006.

[30] D.C. Snowdon et al., "Koala: A platform for OS-level power management," in *EuroSys*, 2009.

[31] X. Wang and M. Chen, "Cluster-level feedback power control for performance optimization," in *HPCA*, 2008.

[32] Y. Yu et al. "DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language," in OSDI, 2008.