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

John D. Davis
johndavis@gmail.com

Suzanne Rivoire
Sonoma State University
Rohnert Park, CA, USA
rivoire@sonoma.edu

Moisés Goldszmidt
Microsoft Research–Silicon Valley
Mountain View, CA, USA
moises@microsoft.com

***Abstract*—Star-Cap is a high-fidelity, real-time power management system for 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 a low-power cluster. 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, even one based on real physical measurements. 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 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.**

I. Introduction

Current data centers cost about US$10 million per megawatt to build, and approximately 40% of the total cost of ownership is related to energy [1]. Active server power management can enable high and efficient utilization of this expensive power infrastructure and at the same time avoid the possibility of causing a catastrophic failure. Unfortunately, current commercial server power management solutions involve additional hardware and software [2], [3], which can double the cost of a server over its 3–5 year lifespan.

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 node-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 [4].

Star-Cap also allows the power caps for individual nodes to adapt to the demands of the workload. Large-scale applications generally allocate different functions to nodes in the rack for locality, matching 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 used to improve efficiency while reducing data center power infrastructure costs. The rest of this paper is organized as follows. We address related work in Section II. Section III describes our clusters, workloads, measurement infrastructure and modeling technique. Section IV presents our power capping algorithm and software system, and Section V evaluates this system. We conclude with Section VI.

II. Related Work

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 [5], but they did not propose specific mechanisms for doing so. Power capping 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. [6]. Because our power models use more features that contribute to dynamic node-level 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 [7]–[12]. These approaches may provide better guarantees of performance and stability, but at the price of increased complexity.

Most previous power capping studies have used either measured power [8], [9] or simple models based on CPU frequency and/or utilization [6], [7], [13]–[15]. Most previous work has used linear models [5], [16], [17] or models that do not capture interaction between predictors [13]. Cochran et al. [18] apply more sophisticated machine learning techniques but base their models on more intrusive hardware performance counters and only examine individual nodes. Our work is the first to apply more comprehensive high-level models to capture the systems' full dynamic range and account for node-to-node variability [19], [20].

Most previous work in power capping has used processor frequency state as the main or only actuator [6], [7], [9], [11]–[14], [21], [22]. We also use DVFS to demonstrate Star-Cap's 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 application-level parameters to trade off performance and QoS or power [23], core parking, component-level low-power states [24], application-level throttling [23], low-power hard drive modes [25], and low-power networking modes [26].

III. Methodology

Star-Cap relies on accurate power models, which we build using only OS-level performance counters as predictors. This section explains our approach to power modeling and describes the workloads and hardware platforms on which we evaluate Star-Cap. The information in this section is condensed from Davis et al. [20].

A. Power models

We begin by running a set of training benchmarks while collecting over 200 OS-level, hardware-independent counters. We use a stepwise regression to reduce this feature space to a handful of the most significant features, specific to the given hardware platform. For the low-power platform we evaluate in this paper, the features are

- Memory: cache faults/sec
- Memory: pool nonpaged allocations
- Disk: total disk time %
- Processor utilization
- File system pin reads/sec
- Total peak page file bytes
- Processor 0 frequency (all cores used the same frequency)

To predict power consumption, we fit a piecewise quadratic function of these OS-level performance counters. The piecewise quadratic model captures two important intuitions:

Piecewise A feature such as CPU utilization may increase the total system power at different rates in different regions of operation. Piecewise models allow us to separate a feature's values into intervals with different coefficients.

Quadratic Quadratic models capture interaction between predictors, rather than modeling power as a linear combination of individual predictors. This is particularly useful in capturing the relationship between CPU frequency and other predictors.

More formally, we use a function $\hat{P}$ of the high-level performance counters, which are represented by $(x_1, \ldots, x_n)$. The form of this model is:

$$\hat{P}(x_1, ..., x_n) = a_0 + \sum_i \sum_j a_{i,j} \cdot B_i^s(x_i, t_i) \cdot B_j^s(x_j, t_j) \quad (1)$$

Fitting the model requires finding the following parameters:

- The coefficient $a_{i,j}$ for each pair of performance counters (i, j)
- The set of knots t_i for each performance counter i
- The signs B_i^s (positive or negative) for each basis function.

In this quadratic model, the parameter s can be a positive $(+)$ or negative $(-)$ sign, and the basis functions B_i^s are hinge functions. $B_i^+(x,t)$ takes a value of 0 if $x \leq t$ and a value of $x-t$ otherwise. Similarly, B_i^- takes a value of 0 if $x > t$ and a value of $t-x$ otherwise. The t thresholds are called *knots*, and a feature can be responsible for multiple knots. The MARS algorithm [27] selects the knots and fits the parameters, which determine how the basis functions interact. We restrict this interaction to degree 2.

We use Friedman's Multivariate Adaptive Regression Splines (MARS) algorithm to automatically fit the quadratic model parameters from training data [27].

Overall, this model provides a balance between low overhead (less than 1% CPU utilization) and high accuracy in the presence of node-to-node variability [19].

TABLE I
Cluster node details

	Mobile cluster	Server cluster
Processor	Intel Core 2 Duo 2-core 2.26 GHz	AMD Opteron 2 × 4-core 2.0 GHz
DRAM	4 GB DDR3-1066†	32 GB DDR2-800
Disk	1 Micron SSD	2 × 10K RPM SATA
Power Range	25–46 W	135–190 W
CPU TDP	25 W	50 W
OS & Filesystem	Win2K8 Server R2, NTFS	

† Maximum DRAM capacity

B. Hardware and software infrastructure

We trained power models and deployed the Star-Cap system on the five-node homogeneous clusters described in Table I. We only present results in Section V 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 [28], [29]. Some workloads are CPU-intensive, while others are dominated by disk or network activity. 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[1], a corpus of about 1 billion web pages. PageRank has high network utilization.

Primes. 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 50 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.

[1] Available at http://lemurproject.org/clueweb09/

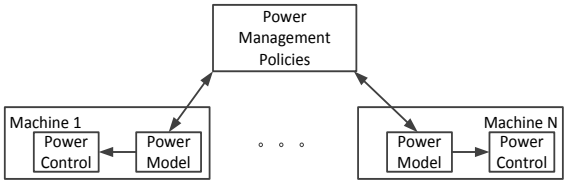

Fig. 1. Two-level power management architecture

1) Hardware: Every node in every cluster is instrumented with a WattsUp? Pro digital power meter and reads its own power measurements over USB. The power meters capture power at 1 Hz and have an error of 1.5%.

2) Software: Each node 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.

IV. Star-Cap: software-only 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.

A. 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 node 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 nodes 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 (node) 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 node-level power capping software must monitor the node'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, 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 inform the selection of knobs to adjust to reach the power management target.

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

B. Control mechanisms

All of the control algorithms we implemented use the same method of adjusting the node-level power consumption, and they all share three common inputs:

- A node-level 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 ($\{f_0, ..., f_k\}$), as described below.

In all three of our node-level control algorithms, the power capping software is responsible for adjusting the power consumption by restricting or increasing the range of available processor frequency states. If the processor supports a set of frequency states $\{f_0, ..., f_n\}$, then the power capping software will restrict the OS to frequency states f_0 through f_k, where $k \leq n$. In our implementations, 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 within a node.

C. Control algorithms

All of the control algorithms we implemented use 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 $\{f_0, ..., f_k\}$ to $\{f_0, ..., f_{k-1}\}$.
- If $P_{current} < P_{lo}$, adjust the available frequency states from $\{f_0, ..., f_k\}$ to $\{f_0, ..., f_{k+1}\}$.[2]

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

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 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.

1) 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 node power to settle; we empirically found that a window of 10 seconds was necessary to fully observe the change of frequency state's effects on the node's power consumption.

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

[2]The direction of this inequality has been corrected from $>$ to $<$ after the publication of this paper.

M-ReCap: $P_{current}$ is the actual, measured node 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 III-A.

2) ProCap details: Like ReCap, ProCap checks $P_{current}$ once per second. However, before adjusting the available frequency states, it uses 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.

V. Evaluation

We report results on the Intel Core 2 Duo cluster, since it has a relatively large dynamic power range and similar per-core power to low-power server platforms. Using QPMs for the server cluster yielded similar results, and 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 [20].

Our dual-core processor has four frequency states: 1596 MHz (70% of peak), 1862 MHz (82%), 2128 MHz (94%), and 2261 MHz (100%). Figure 2 shows the measured power profiles of each node for the four workloads in this study at each of the four frequency states. Altogether, the power consumption ranges from 36 to 44 W per node, or 180 to 220 W for the cluster. Figure 2 also shows the node-to-node power variation for these workloads. Overall, Star-Cap has an overhead of approximately 1% CPU utilization on this platform and lower overhead on more traditional server processors.

A. Power capping profiles

The following figures illustrate our results (a) that accurate models are required for fine-grained management of node or 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 node-level power cap.

1) Low power cap results: Figure 3 shows the results of capping power at a value of 38 W per node for the four workloads. Figure 3 (top) 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 (bottom) shows M-ReCap compared to the proactive power capping algorithm (ProCap). ProCap has fewer power cap violations, with violations of shorter duration.

Figure 3 (top) 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.

2) 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 node 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 node, so the model is unable to accurately predict the node power at high frequency and utilization.

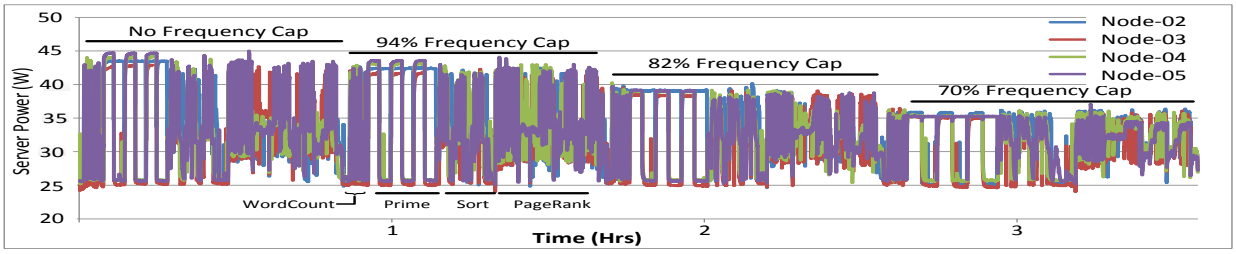

Fig. 2. Workload power characteristics on each node at different frequency caps.

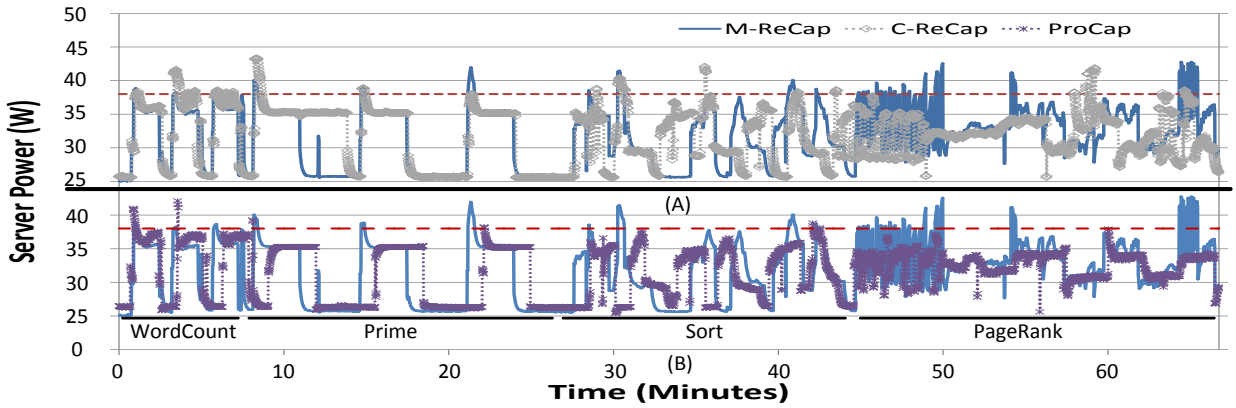

Fig. 3. 38W per-node power cap enforced using (A) reactive capping using measured power (M-ReCap) and cluster-specific power model (C-ReCap), and (B) M-ReCap with proactive capping a using cluster-specific power model (ProCap).

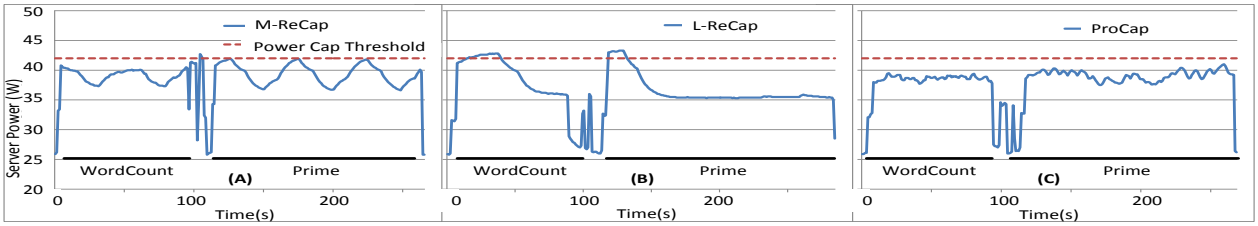

Fig. 4. 42W per-node power cap enforced using (A) reactive capping using measured power (M-ReCap) and cluster-specific power model (C-ReCap), and (B) M-ReCap with proactive capping a using cluster-specific power model (ProCap).

B. Power cap violations

The cluster power cap violations are generally at least 50% lower than the per-node violations, demonstrating that the node violations are not coincident for these workloads, even though the tasks run on these nodes 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 avoid violating 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 provided viable alternatives to static frequency capping that is 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 capping CPU frequency of the node to guarantee a power cap.

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.

C. Power capping and performance

Table II provides the normalized performance of various scenarios compared to running the workloads without a power

TABLE II
NORMALIZED RUNTIMES OF THE WORKLOADS WITH: NO POWER CAP; FIXED PROCESSOR FREQUENCY; A 190W CLUSTER POWER CAP, UNIFORMLY DISTRIBUTED; AND A 190W CLUSTER POWER CAP, NON-UNIFORMLY DISTRIBUTED.

	PageRank	Prime	Sort	WordCount
No power cap (220W)	1	1	1	1
MHz cap: 1596	1.34	1.43	1.22	1.25
190W cap (uniform)	1.20	1.42	1.24	1.27
190W cap (non-uniform)	1.08	1.00	1.08	1.04

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 II 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 nodes, for a per-node target of 38W. This low cap forces the nodes 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 nodes [5].

VI. 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 [30], 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 nodes and increasing the power cap at the granularity of a single node. To the best of our knowledge, this study is the first to apply high-level, full-node 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.

REFERENCES

[1] L. A. Barroso, J. Clidaras, and U. Hölzle, *The Datacenter as a Computer: An introduction to the design of warehouse-scale machines*, 2nd ed. Morgan and Claypool, 2013.

[2] Hewlett-Packard, "HP Insight Control quickspecs," Online, June 2014. [Online]. Available: http://www8.hp.com/h20195/v2/GetDocument.aspx?docname=c04123391

[3] M. Broyles, C. Francois, A. Geissler, M. Hollinger, T. Rosedahl, G. J. Silva, M. Vanderwiel, J. Van Heuklon, and B. Veale, "IBM EnergyScale for POWER7 processor-based systems," Online, 2013 March.

[4] J. Park, "Open Compute Project: Data center v. 1.0." [Online]. Available: http://www.opencompute.org/assets/Uploads/DataCenter-Mechanical-Specifications.pdf

[5] X. Fan, W. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proc. 34th Int. Symp. Computer Architecture (ISCA)*, Jun. 2007.

[6] P. Ranganathan, P. Leech, D. Irwin, and J. Chase, "Ensemble-level power management for dense blade servers," in *Proc. 33rd Int. Symp. Computer Architecture (ISCA)*, Jun. 2006.

[7] M. E. Femal and V. W. Freeh, "Safe overprovisioning: Using power limits to increase aggregate throughput," in *Proc. 4th Int. Workshop Power-Aware Computer Systems (PACS)*, Dec. 2004.

[8] X. Fu, X. Wang, and C. Lefurgy, "How much power oversubscription is safe and allowed in data centers?" in *Proc. Int. Conf. Autonomic Computing (ICAC)*, 2011.

[9] C. Lefurgy, X. Wang, and M. Ware, "Power capping: A prelude to power shifting," *Cluster Computing*, vol. 11, no. 2, pp. 183–195, 2008.

[10] T. Li and L. K. John, "Run-time modeling and estimation of operating system power consumption," in *Proc. Int. Joint Conf. Measurement and Modeling of Computer Systems (SIGMETRICS)*, Jun. 2003.

[11] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No "power struggles": Coordinated multi-level power management for the data center," in *Proc. 13th Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2008.

[12] D. C. Snowdon, E. Le Sueur, S. M. Petters, and G. Heiser, "Koala: A platform for OS-level power management," in *Proc. 4th ACM European Conf. Computer Systems (EuroSys)*, 2009.

[13] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," in *Proc. Int. Joint Conf. Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2005.

[14] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy, "Optimal power allocation in server farms," in *Proc. Int. Joint Conf. Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2009, pp. 157–168.

[15] H. Lim, A. Kansal, and J. Liu, "Power budgeting for virtualized data centers," in *Proc. USENIX Annual Technical Conf.*, 2011.

[16] J. Choi, S. Govindan, B. Urgaonkar, and A. Sivasubramaniam, "Profiling, prediction, and capping of power consumption in consolidated environments," in *Proc. IEEE Int. Symp. Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2008.

[17] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. Bhattacharya, "Virtual machine power metering and provisioning," in *Proc. ACM Symp. Cloud Computing (SoCC)*, 2010.

[18] R. Cochran, C. Hankendi, A. K. Coskun, and S. Reda, "Pack & cap: Adaptive DVFS and thread packing under power caps," in *Proceedings of the 44^{th} Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2011.

[19] J. D. Davis, S. Rivoire, M. Goldszmidt, and E. K. Ardestani, "Accounting for variability in large-scale cluster power models," in *Proc. 2nd Exascale Evaluation and Research Techniques Workshop (EXERT)*, 2011.

[20] ——, "CHAOS: Composable highly accurate OS-based power models," in *Proc. 2012 IEEE Int. Symp. Workload Characterization (IISWC)*, 2012.

[21] S. Govindan, J. Choi, B. Urgaonkar, A. Sivasubramaniam, and A. Baldini, "Statistical profiling-based techniques for effective power provisioning in data centers," in *Proc. 4th ACM European Conf. Computer Systems (EuroSys)*, 2009.

[22] X. Wang and M. Chen, "Cluster-level feedback power control for performance optimization," in *Proc. 14th Int. Symp. High-Performance Computer Architecture (HPCA)*, 2008.

[23] H. Hoffmann, S. Sidiroglu, M. Carbin, S. Misailovic, A. Agarwal, and M. Rinard, "Dynamic knobs for responsive power-aware computing," in *Proc. 16th Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2011, pp. 199–212.

[24] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R. Bianchini, "MemScale: Active low-power modes for main memory," in *Proc. 16th Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2011, pp. 225–238.

[25] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke, "DRPM: Dynamic speed control for power management in server class disks," in *Proc. 30th Int. Symp. Computer Architecture (ISCA)*, 2003.

[26] D. Abts, M. Marty, P. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," in *Proc. 37th Int. Symp. Computer Architecture (ISCA)*, 2010.

[27] J. H. Friedman, "Multivariate adaptive regression splines," *Annals of Statistics*, vol. 19, no. 1, pp. 1–67, 1991.

[28] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed data-parallel programs from sequential building blocks," in *Proc. 2nd ACM European Conf. Computer Systems (EuroSys)*, 2007.

[29] Y. Yu, M. Isard, D. Fetterly, M. Budiu, Ú. Erlingsson, P. K. Gunda, and J. Currey, "DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language," in *Proc. 8th USENIX Symp. Operating Systems Design and Implementation (OSDI)*, 2008.

[30] L. A. Barroso, "Warehouse-scale computing: Entering the teenage decade," 2011, plenary talk, Federated Computing Research Conference (FCRC).