Unleash Stranded Power in Data Centers with RackPacker

Lakshmi Ganesh*, Jie Liu†, Suman Nath†, Feng Zhao†
*Dept. of Computer Science, Cornell University, Ithaca, NY 14853
lakshmi@cs.cornell.edu
†Microsoft Research, One Mirosoft Way, Redmond, WA 98075
{liuj, sumann, zhao}@microsoft.com

Abstract

Data center infrastructures are highly underutilized on average. Typically, a data center manager computes the number of servers his facility can host by dividing the total power capacity of each rack by an assigned "peak power rating" for each server. However, this scheme suffers from the weakness of all static provisioning schemes – it does not account for the variability of load on the servers. We propose an algorithm that studies the power consumption behavior of the servers over time and suggests optimal ways to combine them in racks to maximize rack power utilization. Our algorithm – RackPacker – smooths aggregate rack power utilization by grouping together servers that are unlikely to peak together. Our evaluation of RackPacker on data from an MSN Messenger deployment shows substantially superior results than static packing.

1 Introduction

Modern data center infrastructure, excluding the IT equipment they host, can cost hundreds of millions of dollars to construct. A majority of this cost is attributed to the electrical and mechanical system, which distributes power and cooling to servers, storage, and network devices. Designing data centers to maximally utilize their capacities is therefore a crucial architectural concern.

The capacity of a data center is defined in many dimensions: power, water, cooling, space, network bandwidth, etc. Running out of resources in any of these dimensions means that the service provider needs infrastructure expansion, such as building or renting other facilities. Among these resources, power is usually the first to be exhausted because of the load limitation on the power grid and the increasing power density of computing. However, recent studies [8, 4] have found that on average data center's power resources are highly underutilized.

Salient factors that lead to under utilization of provisioned power are overly-conservative estimates of server power needs, combined with overlooking the dynamic nature of load on servers. The common practice of power provisioning in data centers relies on estimated *nameplate* power consumptions published by the server vendors, which indicates the maximum possible power consumption of the server. In reality, the nameplate power allocated to a server is almost never fully used. To overcome the problem, researchers have proposed to use discounted power rating, obtained by profiling for power provisioning [5]. Static power provisioning, even with discounted power rating, can still leave a large amount of power stranded for two reasons. First, server power consumptions change due to the load fluctuation. Slow and quasi-periodic load fluctuation has been observed in a lot of web traffic, including web sites [1] and instant messaging [3]. This fluctuation can become even more significant as idle power consumption is decreasing for newer servers. Secondly, in addition to the slow fluctuation, there are spikes, caused by short term load variation such as scheduled processing intensive tasks or flash crowd visitors. The discounted power rating – being a worst case estimate – must include both the peak of the fluctuation and the highest spikes; thus it can be overly conservative.

These observations motivate us to design RackPacker, a data-driven approach for power (over-)provisioning, which statistically guarantees that over-subscribed sets of servers do not exceed rack level power caps with high probability. RackPacker takes advantage of two dynamic properties of actual server power traces: (1) Not all servers fluctuate to the peak at the same time. The usage patterns of on-line services can be diverse. If we can bundle services that are maximally out of phase, then the peak of the sum is less than the sum of the peaks. (2) Servers that are managed by the same load balancer or have close dependencies can have strong correlations among their spikes. Statistically, placing services that are *anti-correlated* will lead to smaller probability of their seeing simultaneous spikes. We verify the effectiveness of RackPacker through real workload trace collected from more then 800 servers from Windows Live Messenger cluster.

Note that RackPacker is not a power capping solution. In fact, it relies on power capping techniques, such as those proposed by Ranganathan et. al. [9] or Wang and Chen [10], to provide the safety net in rare events when total rack power exceeds the cap.

2 A Running Example

Throughout the rest of the paper, we use 831 servers from a production Messenger service deployment as a running example for our discussion. Functionality-wise, these servers largely belong to three categories, which we call Types 1, 2, and 3. They are divided into several clusters, where each cluster is managed by a load balancer. Server workloads show strong correlations, because of both functionality dependencies and load balancing effects. For example, when there is a flash crowd, servers behind the same load balancer experience a power spike at the same time, while servers across load balancers are less correlated. Due to the nature of the application, we also observe that about 2 hours after servers of type 1 reach their peak workload, servers of type 3 reach their peak. In addition to the tight coupling among server tiers, the relatively high CPU utilization, reaching over 75% at peak load, make this a challenging set of servers for rack packing.

These servers have a nameplate power rating of 350W; based on this number, a 11.2KW rack can host 32 servers. In other words, we need 26 racks to host these servers in the most conservative situation.

3 The RackPacker Approach

RackPacker takes a data-driven approach that uses collected power consumption traces to support server placement decisions. We assume that services are hosted by virtual machines, even though there may be only one VM per physical server. VMs enable fast re-positioning of services without moving the servers physically. This allows the server placement decisions to be made frequently – at a weekly or even daily basis– and aggressively. The RackPacker algorithm, thus, only needs to predict short term traffic growth. Further, we assume that a power capping mechanism is in place to ensure power consumption never exceeds capacity; in the (probabilistically guaranteed) rare event that power consumption approaches capacity, the power capping mechanism kicks in.

By profiling or monitoring a server operation, we model the server power consumption with a time series (rather than a single number). Figure 3 shows a power consumption trace of a server over a week. The data are sampled every 30 seconds. From the trace, we can see clearly quasi-periodicity that is correlated with the number of users in the system.

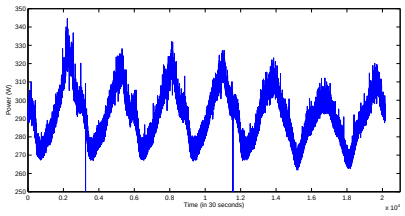


Figure 1. A power trace of a server over a week.

The time series is first filtered to obtain the low frequency power consumption baseline, and the high-frequency noise that captures spikes. The noise signal has zero mean. Its variance represents how "spiky" the transient power consumption can be. The goal of obtaining the low-frequency components is to identify the baseline fluctuations reflecting workload trends, specifically their phase. Using this phase information, we can sift through the servers and bundle those that are most out of phase. The bundles are then treated as the unit for rack packing. The high-frequency noise goes through a covariance analysis that measures the likelihood that two bundles may have spikes at the same time. This statistical measure, together with the baseline of the bundles is used in a statistical bin packing algorithm to find a (sub-)optimal server placement solution.

Thus, RackPacker has three major steps: filtering, bundling, and packing. In the rest of this section, we describe each of these steps in detail.

3.1 Filtering and Classification

The goal of filtering is to separate workload trends from noisy transients. A typical approach is to compute a moving average with a sliding window on the power traces, which is equivalent to low-pass filtering. Let S be the set of servers of interest, P_s be the power profile time series of server $s \in S$ with M samples, and T be the sliding window size to compute the moving average. Then, the baseline B_s is computed as $B_s(i) = \frac{1}{T}\sum_{j=(i-T+1)}^{i} P_s(j), i = \{1...M\}$ (with patching zeros when $i \leq T$), and noise $N_s = P_s - B_s$.

To obtain and compare the relative times at which different servers peak, we perform discrete Fourier transform (FFT) on the baseline signal. In particular, since the most significant fluctuation has the period of a day, we expect that the second FFT coefficient has the largest magnitude. Indeed, for the power profile of the servers we studied, the normalized magnitude of the first 10 FFT coefficients are $[0, 4.2790, 0.2240, 0.7166, 0.4953, 0.1057, 0.1303, 0.0738, 0.0393, 0.0609]$. It is clear that the second component is at least an order of magnitude greater than other components, indicating that it is a good approxima-

tion of the overall shape of the power profile.

We denote the second FFT coefficient of the baseline power profile by f_s. Note that f_s is a complex number that represents a sine wave that can be written as $|f_s|Sin(\omega t + \phi_s)$, where $|f_s|$ is the magnitude and ϕ_s is the phase. In a slight abuse of terminology, we call ϕ_s the *primary phase* of the service.

Based on the relative magnitudes of the noise level and the fluctuation $|f_s|$, the servers can be classified as *flat* or *fluctuating*. Intuitively, a fluctuating server shows substantial load variation above and beyond its noise. In our example, we consider servers whose power profile has $|f_s| < 3\sigma_s$ to be flat. By this definition, 830 out of the 831 servers fluctuate. Fluctuating servers that show significant phase difference will potentially pack well together, and deserve special attention. This brings us to the bundling step.

3.2 Bundling

The goal of bundling is to find small sets of servers whose primary phases "match". Ideally, if the average of f_s across all servers is 0, then the fluctuations cancel each other out . However, in real data centers, this may not be possible. Therefore, the total power load fluctuates at the data center level. Let $\bar{\phi}$ be the average phase of all f_s. Then the best packing approach should spread the data center peak load evenly to all racks. Hence, the target for the bundling process is to make the average phase of each bundle as close to $\bar{\phi}$ as possible.

Another benefit of a common phase for all bundles is dimension reduction. As stated earlier, given a set of power profile time series, we need to verify that at each time instance the total power consumption at each rack does not exceed the power cap with high probability. When server power profiles show distinct phases, we need to perform this verification at the peak time of every power profile. By bundling servers into common phase groups, we only need to verify the time instance when the common phase sine wave reaches the peak.

The bundling process can be explained using complex vectors. The complex coefficient f_s of server s can be viewed as a vector in the complex coordinates, as can the average vector $\bar{f}$ with phase $\bar{\phi}$. Then each vector can be decomposed by projecting it to the direction of $\bar{f}$ and to the direction that is orthogonal to $\bar{f}$. Let f_1 be the 2^{nd} FFT coefficient of server 1, and $\bar{f}$ be the average vector across all servers. Then we project f_1 on $\bar{f}_s$ to obtain $\bar{f}_1$, and then $\tilde{f}_1 = f_1 - \bar{f}_1$. If there exists f_2, whose projection $\tilde{f}_2$ on the direction that is orthogonal to f_s satisfies, $\tilde{f}_2 + \tilde{f}_1 = 0$, then bundling server 1 and server 2 together achieves the common phase. Once common phase bundles are created, further bundling can be performed along the $\bar{f}$ direction so that positive and negative magnitudes cancel each other out .

3.3 Packing

Once bundles are created with the same phase, the packing process uses a modified bin packing algorithm for the final placement. A particular challenge that the packing step addresses is the correlations among the spikes.

The goal of the packing phase is to assign bundles to racks in such a manner as to minimize the probability of exceeding the rack power cap. In order to minimize this probability, the packing phase packs together bundles that show minimal correlation in their spikes (noise). Correlated bundles spike in lockstep; this can result in a heightened likelihood of exceeding the rack cap in the event of load spikes such as flash crowds.

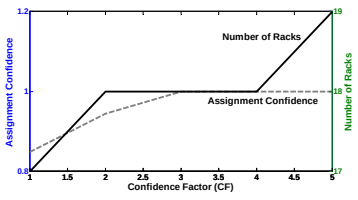
In order to compute sets of bundles that show minimal noise correlation, the packing phase proceeds as follows. First, the bundles are ordered in descending order of size. Bundle size for a bundle b is computed as $\sum_{s \in b} B_s + CF * \sigma_b$, where σ_b is the standard deviation of the bundle noise, and CF stands for confidence factor, a configuration parameter (3, here).

We then iterate through this ordered list of bundles and assign them to racks one by one. A bundle b is deemed to fit into a rack r if $\sum_{b' \in r} B_{b'} + B_b + CF * \sigma_{r,b} < C_r$, where $\sigma_{r,b}$ is the standard deviation of the rack noise (= sum of the noise of each bundle in that rack) combined with the noise of bundle b, and C_r is the rack cap. Given a non-empty rack r, to arrive at the next bundle that we'll attempt to pack into r, we order the unassigned bundles in ascending order of their covariance with the current contents of r. We then try to find a bundle from this ordered list that will fit into r. If no such bundle is found, we create a new rack and repeat the process.

4 Evaluation

Parameter	Value
Rack Cap	11200 W
Bundle Cap	1120 W
ϵ_B	20
Confidence Factor (CF)	3

(a) RackPacker Configuration Parameters



(b) Choice of Confidence Factor

Figure 2. Simulation parameter choices.

We have implemented RackPacker in MATLAB. Fig-

ure 2 shows our choice of parameters for the implementation. The choice of the parameter "Confidence Factor (CF)" is illustrated in figure 2. Here assignment confidence is computed as the percentage of racks that fail to stay within the rack cap over a week's trace of data. We see that the choice of the CF value results in a tradeoff between assignment confidence and packing efficiency.

In evaluating RackPacker, we wish to answer the following questions:

1. *How does RackPacker compare with the prevalent server assignment algorithms?*

2. *What kinds of workloads is RackPacker best suited for? Conversely, are there workloads for which RackPacker is not suitable?*

This section addresses these questions.

4.1 RackPacker: Comparative Performance

To compare the efficacy of RackPacker against current solutions, we use the following metrics:

- **Stranded Power:** This is the difference between provisioned power and actual power consumed per rack. The less the stranded power per rack, the better the server assignment algorithm.

- **Packing Efficiency:** This is the number of racks needed to host the given set of servers. The smaller this number, the better the data center utilization.

We compare RackPacker with two flavors of static assignment: (1) Nameplate Rating-Based assignment, and (2) Peak Power-Based assignment. Both these schemes employ striping, where each type of server is distributed uniformly across all the racks. The *nameplate rating-based scheme* uses the power rating on the server as a measure of its power consumption. Since this number is usually a substantial over-estimate, we also provide a comparison point called the *peak power-based scheme*, which uses the measured peak power consumption of the server in place of the nameplate rating. This is the most aggressive static power provisioning approach, which assumes that the peak in the future does not exceed the peak in the past. In the graphs that we present, the algorithm labelled "Static" refers to the peak power-based scheme.

We evaluate each of these three server assignment algorithms on real power consumption data obtained from a production data center. The data spans 831 servers and hosts the MSN Messenger application. The servers belong to one of three types, corresponding to different tiers of the application. Table 1, and figure 3 describe the data. The data spans a week, but we train the various algorithms on one

Number of server types		3
Number of servers	Type 1	329
	Type 2	283
	Type 3	219
	Total	831
Average power consumed	Type 1	199.4 W
	Type 2	194.7 W
	Type 3	210.1 W
Peak power consumed	Type 1	268.8 W
	Type 2	262.6 W
	Type 3	270 W
Data timespan		1 week
Data time granularity		30 s

Table 1. MSN Messenger Trace Characteristics

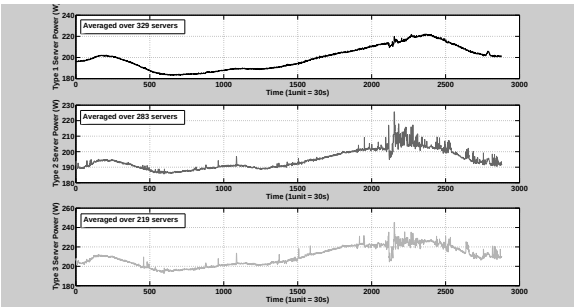


Figure 3. Average Power Consumption Behavior For The Different Server Types

day's data, and validate the computed assignment against the remaining days. An assigment is deemed valid if it causes no power capping incidents. Note that packing aggressiveness can be tuned by tuning this notion of a valid assignment. We had collected over six months' worth of data, but we noticed clear diurnal patterns with each day being largely self-similar, and weekends showing slightly lighter load. Hence the week's worth of data we use in our evaluation is representative of the observed workload.

Figure 4.1 is a pictorial representation of the server assignments computed by RackPacker, and the peak power-based scheme. We find that RackPacker results in 14% more efficient assignment, using only 18 racks against 21 for the peak power-based static assignment. RackPacker does even better when compared with the nameplate rating-based scheme. Recall that using nameplate numbers, we need 26 racks to host these servers. Thus here we see a 30% improvement in packing efficiency.

4.2 RackPacker: Workload Exploration

In the previous section we showed that RackPacker can improve utilization substantially for a real data center scenario. Now we will explore what kinds of workloads Rack-

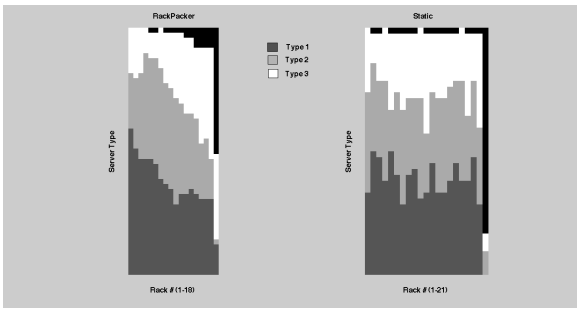


Figure 4. Server assignment results for MSN Messenger trace.

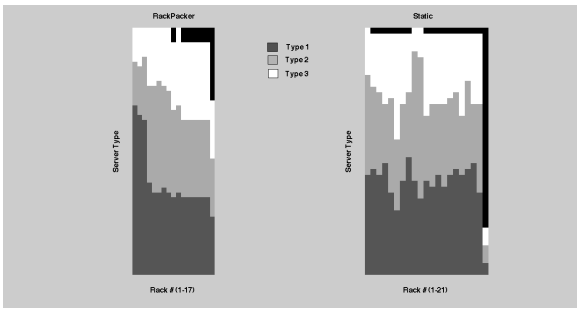


Figure 6. Server assignment results from a workload trace with shifted phases.

Packer is best suited to.

The workload presented in figure 3 represents a single-application hosting center. The three types of servers represent three tiers of the application; we see that these tiers operate essentially in lockstep, with load variation being consistent across the tiers. Here we will explore two other data center scenarios. The data for these scenarios is generated through controlled modification of the real data from table 1.

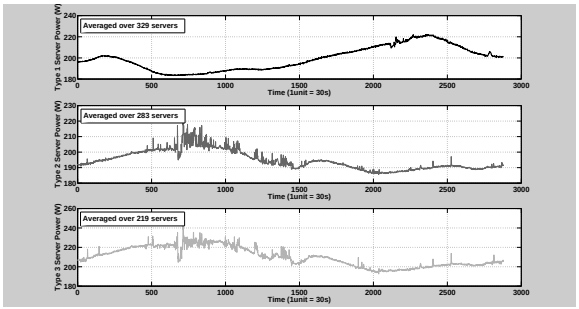


Figure 5. Average Power Consumption Behavior For The Different Server Types

Dedicated Multi-Application Hosting Center: Here we consider data centers that host a small number of applications (more than one). Figure 5 shows the data we generated to represent this scenario. Again, there are three types of servers, but Types 2 and 3 belong to a different application than Type 1 – they are thus phase shifted. Figure 4.2 shows the server assignment computed by RackPacker and the peak power-based static scheme. Again, we find that RackPacker achieves 19% better packing efficiency, using 17 racks against 21 for the static scheme. The nameplate rating-based scheme needs 26 racks (as computed above); RackPacker is now 34% more efficient. In general, we expect that phase shifted servers will benefit more from RackPacker.

Mixed Hosting Center: Here we consider data centers

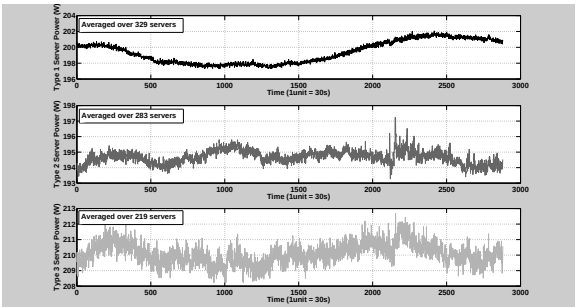


Figure 7. Average Power Consumption Behavior For The Different Server Types

that host a very large number of applications; this represents the cloud computing scenario, where the servers are leased out to various companies that host different applications on them. Figure 7 shows the data we generated to represent this scenario. Here we see that there are numerous types of servers, and their correlations are less obvious. Figure 4.2 shows the server assignment computed by RackPacker and the peak power-based static scheme. Again, we find that RackPacker outperforms the static schemes substantially.

5 Related Work

In this paper, we present a scheme for intelligent oversubscription of data center power. The idea of power oversubscription is not new, and has been explored in the literature in numerous ways. The common theme in prior work, however, is that power tracking/capping are the means used to achieve this oversubscription. To the best of our knowledge, server placement – which sets of servers are placed in which racks – has not been studied as a means of improving data center utilization. Thus, RackPacker is intended to supplement prior work by intelligent server placement that reduces the need for rack-level power capping.

Power capping solutions attempt to provide a means

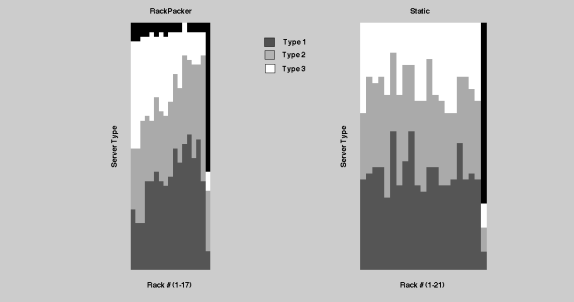


Figure 8. Server assignment results from a workload trace with randomized phases.

to control data center power consumption. This allows data center owners to over-subscribe their facility safely; in the event that power consumption approaches capacity, the power capping mechanism kicks in and prevents it from exceeding capacity. Fan et al [4] show that cluster-level power capping presents a significant power-saving opportunity in data centers through a measurement study. Lefurgy et al [7] present a power capping mechanism that uses a control feedback loop at each server to determine what frequency to run the CPU at. Heath et al [6] add a degree of sophistication to their controller by taking into account the heterogeity of the servers in the data center. Muse [1] is a game-theoretic, distributed power management architecture that tries to allocate only as many servers as are needed to serve the application workload.

Ranganathan et al [9] show the power savings possible through over-subscription at enclosure-level. Their solution uses a controller at the enclosure level that determines the power breakup across the blades, each of which houses an agent to enforce these power budgets. Wang et al [10] present a cluster-level power control scheme that is similar in principle. Our solution takes this principle a step further by determining server placement schemes that maximize over-subscription opportunities.

6 Conclusion

Efficient use of data center infrastructure is a pressing issue for the scalability of the IT industry. Due to conservative and static estimation of server power consumption, traditional approaches for power provisioning leave large amounts of provisioned power stranded. RackPacker is a data driven approach for power provisioning. Our simulation results from real workload traces show that even with tightly coupled and high utilization services, we can achieve significantly better packing performance than static schemes.

An interesting question is how RackPacker compares with random server placement. While an improvement on static packing efficiency may be achieved with a random server placement scheme, we posit that it would take a more structured approach like RackPacker to ensure minimal power capping incidents. This is ongoing work.

Although we implement RackPacker at the rack level, in principle it is applicable at lower (eg, enclosure) as well as higher (eg PDU) levels. Exploring implementation schemes to translate RackPacker to these different levels is an interesting direction for future work.

As a data driven approach for resource management, RackPacker algorithm can be applied to other scenarios, in particular service consolidation via virtualization. The difference is that power is an additive resource, ignoring the power factor, but other resources in a physical server may not be additive. Modeling multi-modality resources and optimizing their utilization is challenging future work.

References

[1] CHASE, J. S., ANDERSON, D. C., THAKAR, P. N., VAHDAT, A. M., AND DOYLE, R. P. Managing energy and server resources in hosting centers. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles* (New York, NY, USA, 2001), ACM, pp. 103–116.

[2] CHEN, G., HE, W., LIU, J., NATH, S., RIGAS, L., XIAO, L., AND ZHAO, F. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation* (Berkeley, CA, USA, 2008), USENIX Association, pp. 337–350.

[3] CHEN, Y., DAS, A., QIN, W., SIVASUBRAMANIAM, A., WANG, Q., AND GAUTAM, N. Managing server energy and operational costs in hosting centers. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems* (New York, NY, USA, 2005), ACM, pp. 303–314.

[4] FAN, X., WEBER, W.-D., AND BARROSO, L. A. Power provisioning for a warehouse-sized computer. In *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture* (New York, NY, USA, 2007), ACM, pp. 13–23.

[5] GOVINDAN, S., CHOI, J., URGAONKAR, B., SIVASUBRAMANIAM, A., AND BALDINI, A. Statistical profiling-based techniques for effective power provisioning in data centers. In *EuroSys '09: Proceedings of the fourth ACM european conference on Computer systems* (New York, NY, USA, 2009), ACM, pp. 317–330.

[6] HEATH, T., DINIZ, B., CARRERA, E. V., JR., W. M., AND BIANCHINI, R. Energy conservation in heterogeneous server clusters. In *PPoPP '05: Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming* (New York, NY, USA, 2005), ACM, pp. 186–195.

[7] LEFURGY, C., WANG, X., AND WARE, M. Power capping: a prelude to power shifting. *Cluster Computing 11*, 2 (2008), 183–195.

[8] LOHR, S. Data centers are becoming big polluters, study finds. In *The New York Times, Technology* (Oct 16, 2008).

[9] RANGANATHAN, P., LEECH, P., IRWIN, D., AND CHASE, J. Ensemble-level power management for dense blade servers. In *ISCA '06: Proceedings of the International Symposium on Computer Architecture* (2006).

[10] WANG, X., AND CHEN, M. Cluster-level feedback power control for performance optimization. In *HPCA '08: Proceedings of the symposium on high performance computer architecture* (2008).