Quantifying Energy-Performance Tradeoffs in Content Datacenters

ABSTRACT

Consolidation techniques for reducing energy use of datacenters typically assume that the performance impact of consolidation comes only from an increase in server load. However, for a common class of datacenters, content datacenters (CDCs), that are used for caching content, user-perceived delay depend significantly on the cache miss rates. In this work, we analyze the energy-performance tradeoff for CDCs while accounting for the increase in server load and miss rate due to consolidation. To this end, we design a system Shrink for load-aware and network-aware consolidation. Using Shrink, we quantify the energy-performance tradeoff based on the workload from a large CDN's datacenter and show that Shrink reduces energy use by 35% over a no-consolidation scheme, while inflating the 95th percentile delay by just 3%. To our knowledge, Shrink is the first system to consolidate servers and switches in a coordinated manner, an approach that reduces network energy use by up to 42% over network-unaware consolidation schemes. Further, we explore content-aware consolidation, which makes content placement and server provisioning decisions to explicitly reduce miss rates. Somewhat surprisingly, we find that content-aware consolidation, even under idealistic assumptions, reduces miss rates over load-aware consolidation at most by 0.06 in the absolute value. Thus, content-aware consolidation may be of limited use to CDCs.

1 INTRODUCTION

Content datacenters (CDCs) used for caching and serving content to end-users are common in today's Internet in which a vast majority of traffic is content such as video and images [10, 22]. CDCs could potentially reduce their energy and cost using consolidation techniques [6, 25]. However, traditionally consolidation is done in a *load-aware* manner based on the implicit assumption that user-perceived performance (or delay) depends only on server load [19, 17]. In contrast, a CDC depends significantly on keeping cache miss rates low to improve user-perceived delay. CDC operators would be wary of deploying consolidation schemes that significantly increase miss rates. Hence, we contend that the tradeoff between energy savings and its impact on user-perceived delay for these schemes is not well understood in CDCs.

Our work sheds light on energy-delay tradeoffs in CDCs by addressing the following key questions. (1) What is the energy-delay tradeoff achieved by load-aware consolidation for real CDC workloads? Does an increase in cache miss rate or an increase in server load play a greater role in determining this tradeoff? (2) Can content-aware consolidation significantly improve cache miss rates over purely load-aware consolidation? (3) Could additional energy savings be achieved in CDCs by network-aware consolidation that coordinates the selection of active servers and switches?

To address these questions, we design and implement Shrink, a load-aware, network-aware and minimally content-aware consolidation system for CDCs. Shrink uses a simple prediction-based load-aware server consolidation and integrates it with randomized load balancing based on content buckets. Shrink is content-aware to a small degree in that it strives to increase effective storage by keeping at most one copy of any content across the datacenter by evicting content from peer caches.

In a large-scale experiment on a public cloud using traces from an Akamai datacenter, we find that Shrink's simple load-aware consolidation reduces server energy use by 35% over a no-consolidation scheme and inflates the mean, 95-th %-ile and 99-th %-ile delays by small margins of of 8%, 3% and 15% respectively over it. Further, we find that at a moderate server load (e.g., 50%), the impact of an increase in both server load and miss rate on user-perceived delay is small. However, at a high load (> 75%), the increase in server load plays a a much greater role than the increase in miss rate in severely hurting user-perceived delay.

Somewhat surprisingly, we find that content-aware consolidation reduces miss rates only by a small margin over purely load-aware consolidation. To evaluate the potential benefit of content-awareness, we design an ideal scheme that handles both the placement of content and the provisioning of servers to minimize the miss rate for a given energy budget. Even with several idealistic assumptions, we find that the absolute difference in miss rates between the content-aware scheme and Shrink's load-aware scheme is at most 0.02 for the Akamai trace and 0.06 for a modified version of the trace with less skew. More importantly, we expect that these small gains in miss rate due to content-aware consolidation would result in even smaller gains in user-perceived delay that depends on other factors (e.g., server load) as well.

We attribute the small incremental benefits of content-aware design to a number of factors. First, we find that caches warm up much quicker compared to the average time a server is active and hence on-off transitions minimally affects total miss rate. Second, real workloads are highly skewed and can achieve low miss rates despite significant reduction in storage due to consolidation. Third, we observe a correlation between CDC load and the content working set size. As a result, load-aware schemes also end up making similar provisioning decisions as content-aware schemes and achieve low miss rates in a similar manner. Overall, these results leads us to conclude that sophisticated content-aware approaches may be of limited value to CDCs.

Shrink's network-aware consolidation appears a promising approach to save energy. While multiple efforts have studied switch

Conference'17, Washington, DC, USA
© YYYY ACM. 978-x-xxxx-xxxx-x/YY/MM...$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

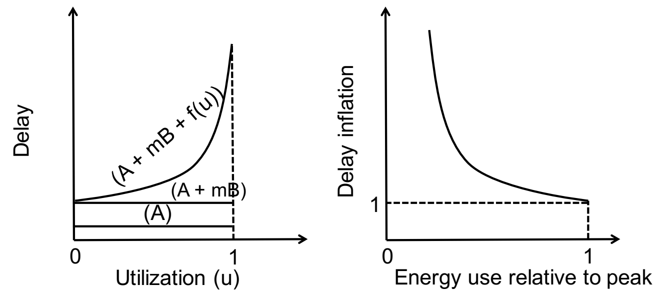


Figure 1: [Left] An illustrative server utilization vs. delay curve. [Right] Corresponding energy-delay curve.

and server consolidation in isolation from one another [33, 15, 37, 7, 4], Shrink uses a simple heuristic that coordinates the two tasks to increase potential energy savings. It selects the active servers in a left-to-right order in a topology, and thereafter selects a subset of active switches so that the average bandwidth from active servers to external clients is the same as in an unconsolidated datacenter topology.

In our experiments using common datacenter topologies including FatTree [2] and VL2 [13]. , the energy use of switches with Shrink's network-aware consolidation is close to a lower bound and up to 42% less than a network-unaware consolidation scheme. In an experiment on Emulab [35], Shrink's network-aware consolidation reduces server and switch energy use by 57% over a no-consolidation scheme while inflating delay by 15% over it. Thus, Shrink can perform both server and switch consolidation with a small delay inflation in CDCs.

2 CDC ENERGY-DELAY MODEL

We present a simple model of a CDC to illustrate that the energy-delay tradeoff depends both on server load and cache hit rates. We use the model to motivate a number of research questions that we seek answers to.

2.1 Single server

Consider a single CDC server serving a workload of content requests from end-users that is unchanging, i.e., the arrival rate, popularity distribution, and size distribution of content across requests is fixed. Let m denote the cache miss rate at the server. Assume that the power drawn by a server $p(u)$ is a function of its utilization u, or the ratio of the incoming load and the server's capacity [19].

The end-to-end user-perceived delay is assumed to be the sum of three components as also illustrated by the three curves in Figure 1 (left): (1) Mean server delay $f(u)$, assumed to be a convex, increasing function of the utilization u ($0 \leq u \leq 1$); (2) Server-to-origin delay B, which is constant but incurred only upon a cache miss; (3) Client-to-server delay A, a constant. Thus, the total delay is $f(u) + mB + A$.

2.2 Datacenter as a single logical server

The simplistic model above can serve as a first-order approximation of a datacenter viewed as a single logical server as follows. Suppose the datacenter consists of N homogeneous servers, each identical to the one above. Suppose the total incoming load is uniformly distributed across all N servers resulting in a fixed utilization U at each server, which implies a total power consumption of $p(U)N$, and a mean delay of $f(U) + mB + A$. We have implicitly assumed here that the miss rate m remains the same as above even though each server is getting a sampled transformation of the original request distribution.

Consolidating the datacenter to n ($< N$) servers while serving the same workload results in following changes:

- Utilization of each server increases to UN/n.
- Total available storage by n/N leading to a higher miss rate $m'(> m)$.

The aggregate power consumption of the CDC becomes $p(UN/n)n$. The corresponding energy use relative to the peak is $\frac{p(UN/n)n}{p(U)N}$. The delay is given by $f(UN/n) + m'B + A$, and the corresponding delay inflation is given by $\frac{(f(UN/n)+m'B+A)}{(f(U)+mB+A)}$. Figure 1 (right) illustrates the curve of relative energy use and delay inflation as per this model.

We make two observations based on the expression for the delay inflation above. First, the more skewed (closer to L-shaped as opposed to linear) the server's utilization vs. delay profile is, the less noticeable the impact on delay as $f(UN/n) \approx f(U)$ unless $UN/n \to 1$. Second, the more skewed (e.g., a high Zipf exponent) the popularity distribution is, the less noticeable the impact on delay as $m' \approx m$ assuming that the consolidated storage also suffices to cache the small fraction of popular objects contributing to the overwhelming portion of hits.

2.3 Open issues

The above simple model is useful for exposition, but leaves a number of open questions that we address in the rest of the paper.

(1) Our model considers a static workload and assumes a datacenter acts a single logical server of larger size. In practice, real workloads have time-varying load and content popularity, and real CDC servers need to deal with non-uniform load distribution and unequal miss rates. How do these factors affect the energy-delay tradeoff?

(2) Existing load-aware consolidation makes no explicit effort at minimizing miss rates. In the ideal case, how much room is there for improving miss rates without sacrificing energy savings?

(3) The above discussion ignores the energy use of network switches in a CDC. How do simple schemes that coordinate server and switch consolidation save CDC energy? How do these scheme compare to existing techniques for switch consolidation?

To answer these questions, we develop consolidation techniques in Sections 3, 4 and 5 and evaluate those techniques using a real workload from an Akamai datacenter in Section 6.

3 CONTENT-AWARE CONSOLIDATION

Content-aware consolidation seeks to reduce miss rates towards improving the energy-delay tradeoff in CDCs. Our work focuses on two features of a content-aware design – *content placement* and *server consolidation.* We present an ideal content placement scheme and an ideal server consolidation scheme to provide an optimistic estimate of the improvement that a content-aware design can achieve.

3.1 Content placement

Content placement technique determines the set of content cached in a CDC and which server each content is stored at. In an unconsolidated CDC, content placement is determined by its caching strategy. Consolidation in CDCs, in addition, creates a unique content placement problem – cache compaction. By default, content stored on the servers being turned off will become unavailable, potentially hurting miss rates. To improve miss rates, a cache compaction strategy selectively copies some of the content from these servers to those servers that will remain on [38]. Further, if cache evictions are necessary to make room for the content being copied, the compaction strategy also determines the set of content to be evicted.

We make a key assumption in designing our ideal content placement scheme. As shown by multiple recent studies using real content workloads, we assume that an LRU caching strategy achieves near optimal miss rates for an unconsolidated datacenter [28, 11]. Our approach below works also with other schemes as long as they have the following *subset property*: for the same workload, a smaller cache will always contain a subset of entries of a larger cache.

Ideal content placement: This scheme treats the CDC as a single logical server with a single unified cache of size equal to the sum of storage on all active servers. The unified cache maintains a single LRU queue and evicts content that is least recently used across all active servers. Therefore, the unified cache should outperform real CDCs in which the cache at each server evicts the locally least recently used content. In practice, the overhead of coordination among servers to maintain a single unified cache would be non-trivial. However, we ignore this overhead since we are interested in an ideal scheme.

For cache compaction from n servers to n' $(< n)$ servers, the LRU cache retains the longest prefix of LRU cache entries that fit in the cache with n' servers. This strategy is ideal for a caching scheme with the subset property. This is because it leaves the cache in the same state as if it were of the smaller size, i.e., n' servers, to begin with. Again, we ignore any overhead on servers in order to achieve this compaction.

3.2 Server consolidation

Our server consolidation seeks to minimize total cache misses given an energy budget for a time window (e.g., a day). To this end, it pre-plans the number of servers provisioned in each interval in that window based on estimated workload. For exposition, let us assume provisioning decisions are made on an hourly basis. Further, we assume that the average load on the CDC and the relation between storage and miss rates for each hour in the day is already known. In practice, these statistics can be estimated by analyzing recent workload, e.g., previous day. The key to our approach is to use these statistics to holistically optimize provisioning across all time intervals to achieve the above objective.

Ideal server consolidation problem: We consider a CDC with N identical servers with a capacity to serve a unit load per time interval and with a unit energy consumption per time interval. Each server is assumed to have local storage for caching content. The workload spans T consecutive time intervals of equal length. The total load in the interval t $(1 \leq t \leq T)$ due to content requests is l_t. If n servers are active in the time interval t, let $m(n,t)$ denote the miss count, i.e., the number of cache misses, for that interval. Below, we leverage the ideal content placement scheme in the computation of $m(n,t)$. For a fixed t, we assume $m(n,t)$ is a convex, decreasing function, i.e., increasing the number of servers in an interval provides diminishing improvements in miss count.

The output is the minimum miss count and the number of servers n_t to be kept active for time t $(0 < n_t \leq T)$ such that following two conditions are satisfied: (1) All load should be served: $n_t \geq \lceil l_t \rceil$. (2) Total energy consumption should be less than a constant E: $\sum_{1 \leq t \leq T} n_t \leq E$.

Solution: The minimum miss count is computed using the following greedy algorithm. The greedy algorithm results in an optimal solution due to the convexity of $m(n,t)$ for any fixed t.

Step 1: Initialize the number of servers to be provisioned in time interval t as $n_t = \lceil l_t \rceil$.

Step 2: While $\sum_{1 \leq t \leq T} n_t < E$, turn on one more server in the interval t that decreases the miss count the most. i.e. $n_t = n_t + 1$ where

$$t : \max_{1 \leq t \leq T} (m(n_t, t) - m(n_t + 1, t))$$

.

The minimum miss count equals $\sum_{1 \leq t \leq T} m(n_t, t)$.

Computing $m(n,t)$: We use the ideal content placement proposed above in computing the function $m(n,t)$. In particular, we assume that the n servers form a unified cache. Starting from an empty cache, we simulate the LRU cache for the entire workload from $t = 1$ to $t = T$. Then, $m(n,t)$ equals the number of misses in each time interval t.

4 NETWORK-AWARE CONSOLIDATION

Unlike existing approaches that treat server and switch consolidation in isolation, a network-aware consolidation coordinates the two to achieve greater energy savings. In this section, we explain our approach using an example and present our heuristic consolidation strategy. Later, we show that the simple heuristic can achieve energy savings close to lower bounds for common topologies.

4.1 A motivating example

We use Figure 2 to show the benefit of coordinating server and switch shutdown strategies over an uncoordinated approach. In that Figure, both the left and the right topologies have 20 active servers. The right topology requires all ToR switches to be kept active. In the left topology, all active servers are placed in the one leftmost rack, which allows ToR switches in other racks to be turned off. This example shows that consolidating servers in a network-aware manner can reduce switch energy use.

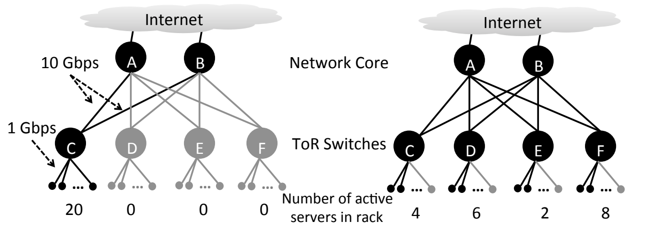


Figure 2: Black (grey) components are turned on (off). Each rack has 20 servers. (Right) Randomly choosing the set of active servers results in all ToR switches being turned on. (Left) Choosing the same number of active servers in a "network-aware" manner enables more ToR switches to be turned off.

What constraint should be enforced so that shutting down switches does not severely impact user-perceived delay? Our constraint is based on the observation that CDCs primarily send and receive traffic from external end-points, such as end-users and origin servers. We require that all active servers should be able to simultaneously send traffic equal to the *external bandwidth* outside CDC using the set of active switches only. External bandwidth is equal to the outgoing link capacity of a server divided by the over-subscription ratio (> 1) of the datacenter. If a server sends traffic up to the external bandwidth, then this condition avoids network bottlenecks for common CDC topologies that are multi-rooted trees.

We explain the above observation using an example. Consider the topology in Figure 2 (left) in which the external bandwidth is 1 Gbps. Let us assume that only servers connected to switch C are active due to consolidation. Let the maximum rate of traffic from a server to clients be 800 Mbps. If all active servers are sending traffic at their maximum rate, then the total traffic from the switch C to the core switches A & B is (800 Mbps)$\times$20 = 16 Gbps. If both A & B are active, the condition above is satisfied, and all servers can simultaneously send traffic to clients at their maximum rate. If A is on but B is off, the condition above is not satisfied, and as a result, a network bottleneck happens on the link AC.

4.2 Network consolidation scheme

We assume a datacenter topology in the form of a multi-rooted tree in which a topological ordering of nodes from left to right is well-defined at each level in the topology. Servers and switches reside at leaf and non-leaf nodes respectively; root nodes provide external connectivity. Links are symmetric and all links at a level have identical capacity. To adapt network routing in response to switch consolidation, we assume datacenter switches support ECMP [16].

Our approach proceeds in two main steps. First, we select the active servers, in a network-aware manner, to be the leftmost leaf nodes in a topology. The number of active servers is determined by the server consolidation scheme described in the next Section. Next, we select the subset of active switches assuming that we need to route traffic equal to the external bandwidth from each active server to the root nodes.

For the second step, we consider switches in the order of increasing height from leaf to root and among switches at the same height considers them in a left-to-right order. If a switch does not receive any traffic from its children, it is turned off. Otherwise, we select its p-leftmost parents that must be active to forward the traffic sent by its children to the root nodes; p is the least value for which the sum of capacities of links to the selected parent nodes is equal or more than than the traffic sent by the switch's child nodes. Assuming ECMP support in switches, traffic forwarded to root nodes is assumed to be divided on links to the p parent nodes equally.

5 SHRINK DESIGN AND IMPLEMENTATION

Shrink is a system for load- and network-aware consolidation in CDCs. It enables us to precisely quantify the energy-delay tradeoff, which is not possible using a simplistic model such as that in Section 2. Shrink implements our network-aware consolidation scheme described in Section 4. This section presents Shrink's server consolidation and load balancing algorithms followed by Shrink's implementation status.

5.1 Server consolidation

Shrink's server consolidation seeks to keep server utilization below a specified limit while limiting the impact on hardware reliability by reducing the rate of on-off transitions of servers. Shrink computes the number of active servers using two integer values: *minServers* and *maxServers*. $minServers = \lceil L/U \rceil$, where U is an operator-specified limit on server utilization and L is the load predicted for the next interval using linear regression based on the total load across servers in the past few intervals (default = 10 intervals). *maxServers* is the maximum value of *minServers* over the previous time window of length W. If $minServers > n$, where n is the current number of active servers, $(minServers - n)$ more servers are turned on, else if $maxServers < n$, $(n - maxServers)$ servers are turned off.

The reliability of servers depends on the parameter W, which we call the pre-shutdown wait interval. With a smaller W, Shrink turns servers off sooner and saves more energy but potentially increases the rate of on-off transitions because of spurious decreases in load. Experiments with our Akamai datasets (Section 6.2) show that W close to 30 min is a sweet spot for which energy use is 15% higher than that achievable for a small value of W = 1 min, and whose on-off transition rate is nearly 10$\times$ lower than that of W = 1 min. Operators can better inform their choice of W with similar tests for their workload.

A low rate of on-off transitions of servers improves switch reliability also. This is due to the following property of our consolidation schemes: if the number of active servers increases, additional servers and switches are turned on, but none of the servers and switches that are already active are turned off. For example, consider a time interval in which the number of active servers is monotonically increased from 1 to N, where N is the total number of servers. In this interval, there is at most one off to on transition for each server. Since Shrink's switch consolidation only turns on new switches upon an increase in the set of active servers, each switch would also have at most one off to on transition in this interval.

5.2 Load balancing

Shrink uses randomized load balancing over a set of content buckets. Content is mapped to a fixed number of buckets (default = 1000) using a consistent hash function based on its name. Shrink computes a *traffic split* for each bucket, which determines the ratios in which the bucket's requests will be distributed among servers. In each iteration, Shrink predicts each bucket's load using linear regression and computes new traffic splits for a subset of buckets. This subset consists of buckets that either belong to servers that are no longer active or whose load exceeds a load limit. This load limit is equal to the the server utilization limit U or the average load on active servers, whichever is greater. To compute new traffic splits for the selected buckets, Shrink assigns the load for a bucket to a server chosen randomly. If the load on that server exceeds the load limit due to this assignment, then the excess load for that bucket from that server is assigned to the least loaded active server.

Shrink seeks to keep at most one replica of any content across the active servers to increase effective storage for caching. If a content is available at peer caches within the CDC but not available locally, it is copied from any peer at which it is available and cached locally. Copying content from peers, combined with frequent changes in traffic splits for a bucket, could create multiple content replicas, thereby reducing effective storage. Hence, after copying content from a peer cache, that content is evicted from other peers, and thus only a single content replica is retained.

5.3 Implementation

Our Shrink protoype is implemented in Java (6K LOC) and runs as a control program that executes at periodic intervals (default = 20 sec). In each interval, Shrink receives load reports from all active servers at the granularity of content buckets. Load is measured in terms of request rates, which is measured from content access logs output by the caching proxy. A client first contacts Shrink to receive the IP of a server (similar to a DNS resolution), and then downloads content from that server. Shrink uses Squid as a caching proxy [30]. We use Squid's asynchronous, multi-threaded storage AUFS. To reduce miss rates, a large file is requested as a sequence of 2 MB chunks, which enables Squid to cache each chunk independently.

There are a couple of limitations of our prototype. First, it does not support querying and fetching content from peer caches inside CDC. We found the overhead of querying all peer caches using Squid's Internet Cache Protocol [34] to be high (nearly 25% of total load). Hence, we evaluate peer caching using simulations only. Second, the prototype lacks power controls for servers and switches. We emulate the startup (shutdown) delay of servers by waiting for a pre-defined interval before restarting (killing) the server-side processes. We note that remote power management for servers and switches is readily available today [36, 27].

6 PERFORMANCE EVALUATION

Our evaluation has the following key goals: (1) Quantify the tradeoff between energy and user-perceived delay achieved by Shrink's load-aware and network-aware consolidation. (2) Evaluate the benefit of content-aware consolidation in reducing miss rates over load-aware consolidation. (3) Compare Shrink's network-aware consolidation against alternatives for reducing network energy use.

6.1 Shrink prototype-based experiments

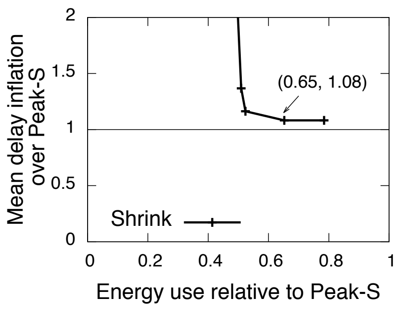
Akamai dataset: Our evaluation uses content access traces from an Akamai datacenter. The traces include all requests received at a datacenter with 18 servers for a week in December 2013. We restricted our data collection to a small datacenter as we did not have the resources to experiment with traces from a significantly larger datacenter. Our anonymized traces include several major types of traffic observed in a CDN such as video, social media and other web traffic. Each anonymized log entry includes among other fields, the request timestamp, content URL, size of requested content, actual number of bytes sent and IP address of the user. Overall, the traces contain more than 2 billion requests generating nearly 200 TB of traffic.

6.1.1 Server consolidation. This experiment quantifies energy-delay tradeoff Shrink's load-aware server consolidation relative to *Peak-S*. Peak-S represents a baseline in which a CDC operator does not use consolidation and keeps the entire datacenter always on. We have conservatively chosen the number of servers used by Peak-S (15 servers) to be less than the number of servers in the Akamai datacenter itself so as not to overestimate the energy savings. We evaluate Shrink for multiple values of server utilization limit U from 0.375 to 0.875. We use a pre-shutdown wait interval $W = 10$ min. Our energy calculations assume that the ratio of the idle to peak energy use of servers equals 0.5 and energy use increases linearly with utilization [5].

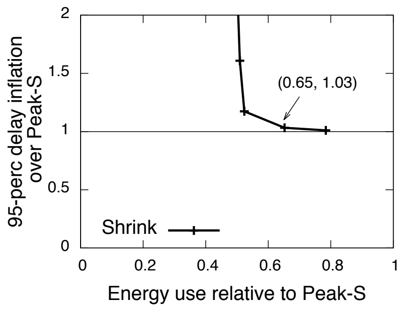
Our EC2 testbed consists of 15 CDC servers, 15 clients and 4 origin servers running on independent m3.xlarge instances (4 core, 15 GB RAM, 40 GB×2 SSD), all in the same datacenter. Our origin server is a trivial Apache Tomcat application that dynamically generates the requested content. We emulate a 60 ms RTT between origin servers and CDC's servers, and a 10 ms RTT between client and server machines. We configured each server to use an 8 GB memory cache and a 30 GB cache on each SSD. Our workload consists of a 24-hour duration of the trace. We selected one-eighth of the content randomly from the trace but sped up the trace by 8× to send those requests over a 3-hour duration. Thus, we maintain approximately the same load on the servers.

Figure 3 compares the delay and the energy use of Shrink relative to Peak-S for the mean, 95-th percentile and 99-th percentile of user-perceived delay. Shrink reduces energy use by 35% over Peak-S while inflating the mean, the 95-th percentile and the 99-th percentile by 8%, 3% and 15% respectively. To explain the difference between Peak-S and Shrink, consider Figure 4 (top) which shows the aggregate load and the number of servers from one of the runs of the system. Shrink adapts the number of active servers based on load in the system keeping only 3 servers active when the load is the lowest, but Peak-S always keeps 15 servers active and hence has a higher energy use.

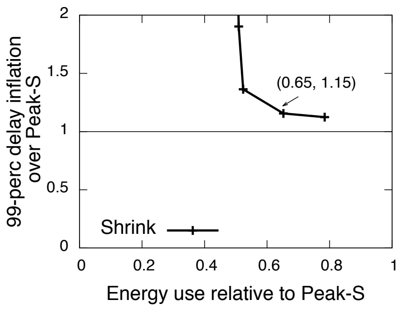
Does an increase in server load or an increase in miss rates cause a greater impact on Shrink's delay over Peak-S? In Figure



(a) Mean



(b) 95-th percentile



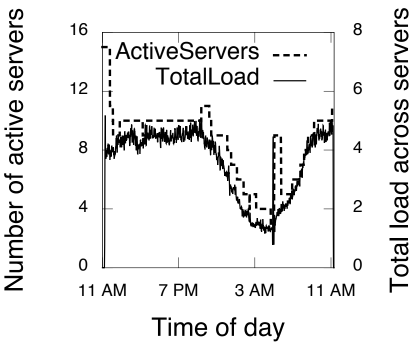
(c) 99-th percentile

Figure 3: [Prototype] Server consolidation (Section 6.1.1): Shrink's load-aware consolidation reduces energy over Peak-S with a small delay inflation. In Figure 3a, Shrink's energy use is 0.65× of Peak-S and its mean delay is 1.08× of Peak-S.

4 (bottom), the x-axis shows the server utilization limit U and y-axes show the corresponding hit rates and the mean delay of Shrink. We find that Shrink's hit rates are lower than Peak-S by 3% to 7%. Thus, the delay inflation due to a decrease in hit rates is likely to be small for all utilization levels. A small reduction in hit rates is not surprising given that the Zipf exponent for the Akamai trace is 0.8 as per our calculations, and our model in Section 2 has suggested that consolidation reduces hit rates by a small fraction for workloads with a high skew in content popularity. We find that mean delay increase sharply at a high server utilization limit, e.g. $U = 0.875$, which is likely due to an increase in server load. To summarize, hit rate reduction has a small impact in user-perceived delay in our experiments with a real CDC trace. But, an increase in server utilization can severely inflate delay, especially at high utilization limits.

6.1.2 Server & switch consolidation. We use Emulab [35] to evaluate Shrink's energy-delay tradeoff when both server and switch consolidation are being performed. Figure 5 (top) shows our experimental topology with 1 Gbps links. Since all switches in the topology have 4 or more ports, we calculate the energy use of switches based on the power use of 4-port switch (Netgear GS105, 14.4 W) [20]. The energy use of servers is computed as in the previous experiment. Our workload is a 1-hour duration of the trace containing requests for one-eighth of the content selected randomly.

Figure 5 (bottom) compares schemes in terms of the mean delay. Across experiments with varying server utilization limits, Shrink uses between 2 and 9 servers; Peak-S uses 9 servers. We discuss the case when Shrink uses 3 servers so that only one of the ToR switches are being used as a result of network consolidation. In this case, the peak utilization of the link between the ToR and the core switch increased up to 76% during the experiment, which is nearly 3× higher than the peak link utilization for Peak-S. Despite this increase, Shrink's delay is only 15% higher than Peak-S, while its switch energy use is 50% lower and the total energy use is 57% lower than Peak-S (second point from the left in Figure 5 (bottom)). This result suggests that Shrink can perform both server and switch consolidation with a small delay inflation in CDCs.



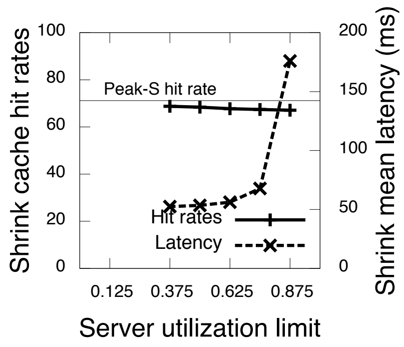
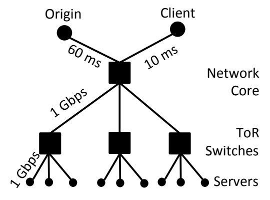


Figure 4: [Prototype] Server consolidation (Section 6.1.1): [Top] Shrink adapts the number of active servers based on CDC load to reduce energy over Peak-S. [Bottom] Cache hit rates and mean delay for Shrink and Peak-S.

6.2 Content-aware consolidation

Schemes compared: We evaluate Shrink by varying U in the range 0.5 to 0.9 and compare it against these schemes.

Ideal-content-aware uses the server consolidation algorithm in Section 3. To run this algorithm, we define the length of each time interval as 5 min, and compute the average request load (in terms of network traffic) in each interval for our workload. For a given number of servers n, the miss rate functions $m(n, t)$ for each



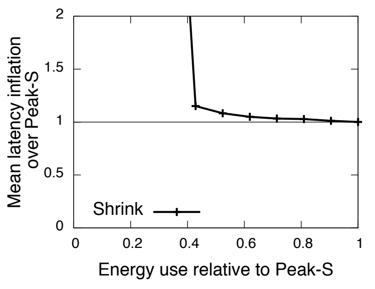


Figure 5: [Prototype] Server & switch consolidation (Section 6.1.2): [Top] Emulab topology for the experiment. [Bottom] Compared to Peak-S, Shrink has a 15% higher delay but a 57% lower energy use.

t are computed by running an LRU cache simulation for the entire workload in this experiment.

Shrink-idealCP uses the same number of servers as Shrink at all times, but it uses the ideal content placement scheme described in Section 3.1. This scheme is useful to understand the difference in performance of Shrink's content placement and the ideal placement in which the storage across all servers is treated as a single logical cache.

Shrink-w/o-peer-eviction does not evict content from peer caches after fetching content from peer caches inside a CDC. It serves to evaluate the benefit of this feature in our design.

Methodology: We conduct trace-driven simulations for two traces: the Akamai trace and for a modified version of that trace with significantly reduced skew in content popularity. Our modified trace dramatically reduces the number of requests for a small number of highly popular content. Specifically, we selected K most accessed content that generate a fraction (=60%) of all requests in the trace. We retain only as many requests for these K content as the number of requests for the $(K + 1)$-th most accessed content, and delete the rest. We replace deleted requests by introducing new requests for other content in proportion to their access counts. To maintain the temporal locality of requests, the new requests for a content are added in the trace within a 1 hour window of the existing requests for that content.

Our experiments define the capacity of a server is defined in terms of its outgoing network link capacity. The rate of network traffic generated by a request is a constant equal to the client bandwidth reported in the Akamai trace. The energy use of a server is proportional to the duration for which it is active. To be able to fit the simulator process in the memory on our machine (32 GB), we filtered requests for one-eighth of the content from the trace. Accordingly, we scale down the link capacity and storage of each server by one-eighth relative to that of the servers in the Akamai datacenters. The cache miss rates of our simulator's LRU caching and Squid differ by less than 2% for the same workload and cache size.

Results: For the original Akamai trace, the absolute difference in miss rates for all schemes, including Ideal-content-aware, was less than 0.02 (i.e., 2% of all requests). The implication being that content-awareness makes a minimal improvement in miss rate over load-aware consolidation for a real CDC workload. Hence, we focus our discussion on the modified workload, where the difference among schemes is slightly more.

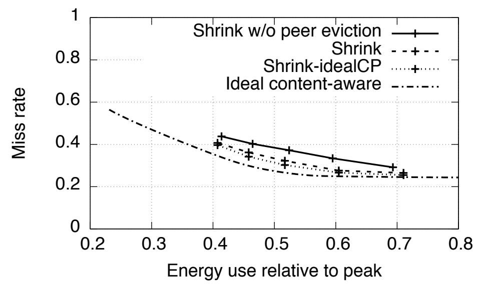


Figure 6: [Simulation] Shrink's load-aware consolidation achieves a miss rate that differs from Ideal-content-aware by at most 0.06 for the same energy use

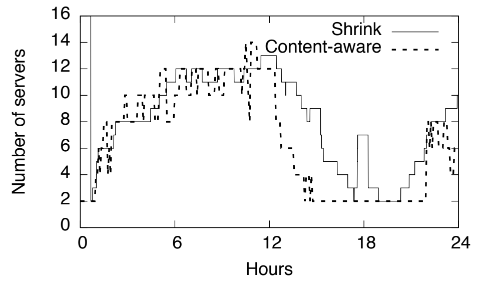


Figure 7: [Simulation] Shrink and Ideal-content-aware provision similar number of servers for a large fraction of time

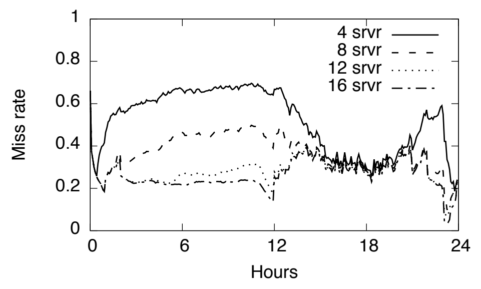


Figure 8: [Simulation] Miss rate computation over a 24-hour period for a constant amount of storage equal to that of 4, 8, 12 and 16 servers

Figure 6 shows the tradeoff between energy use and miss rate for the modified workload. The energy use is normalized with respect to the maximum energy use assuming no consolidation is performed. For the same energy use, the maximum difference in miss rate between Ideal-content-aware and Shrink is at most 0.06 (0.37 vs 0.31). We believe that a practical content-aware scheme is likely to results in a higher miss rate than Ideal-content-aware, e.g., due to a lack of accurate future knowledge of content workloads.

Two key reasons explain why Ideal-content-aware provides only small improvements over Shrink's load-aware consolidation. First, we find that the cache warm up for this workload is of the order of a few (< 10) minutes, whereas the average active duration of the server for Shrink is 6 hours or more. Thus, Shrink quickly regains its hit rate after servers are shut down in a CDC. Second, we find that the number of servers provisioned by Ideal-content-aware and Shrink is nearly equal for a large fraction of the time as shown in Figure 7. An explanation for this finding is that datacenter load and the size of the content working set are correlated. When load is higher (lower), content working set size is larger (smaller) as well and vice versa. As a result, Ideal-content-aware tends to provision more (less) servers at those times when the load is greater (smaller), and ends up making similar decisions as a load-aware scheme.

We show the correlation between load and the content working set using Figure 8, which shows the results of a miss rate computation for storage equal to that of 4, 8, 12 and 16 servers over the same period as in Figure 7. We observe that in the first part of the graph when the load is higher, the size of the working set is larger since 12 servers are needed to significantly reduce miss rates. In comparison, in the second part of the graph where the load is lower, working set size is smaller and just 4 servers are needed to achieve close to the minimum miss rates.

In Figure 6, Shrink-idealCP and Shrink differ in miss rate by at most 0.02. There are two reasons that explain this finding. First, Shrink aggressively evicts replicas of content at peers, which helps increase effective storage for caching content and reduce miss rates. The benefit of peer eviction can be seen in Figure 6, where Shrink-w/o-peer-eviction has a miss rate that is higher by up to 0.06. Second, Shrink's bucketing and load-balancing use randomization due to which individual servers receive a request workload with characteristics similar to the overall CDC workload. As a result, all servers achieve miss rates that are similar and close to the miss rate achieved by an ideal content placement. Overall, this finding shows that Shrink load balancing is highly effective at placing content among the active servers.

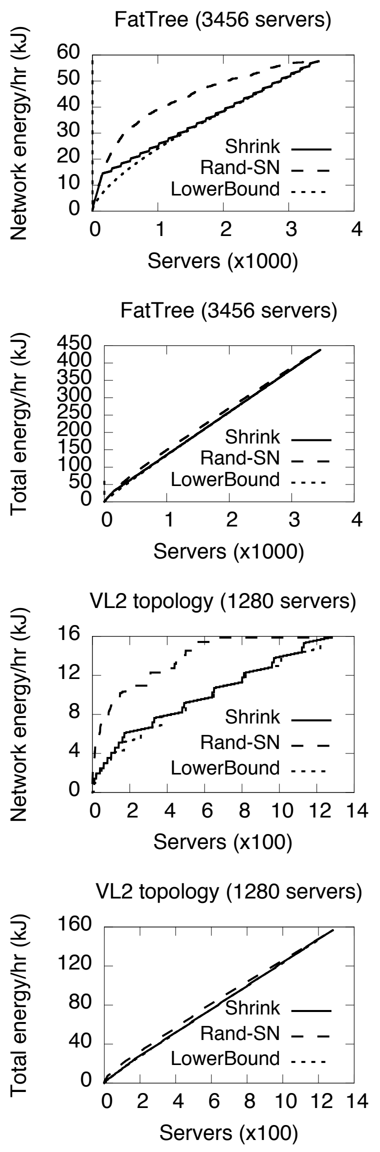


Figure 9: [Numerical computation] Shrink's network energy use is lower than a network-unaware server consolidation scheme Rand-SN by 38% on FatTree and 42% on VL2 when one-fourth of the servers are active in each topology.

6.3 Network-aware consolidation

Schemes compared: We compare the energy use of switches for Shrink and the following two schemes. (1) *Rand-SN:* Unlike Shrink, Rand-SN does not coordinate the selection of active servers and switches. It selects the set of active servers randomly in a network-unaware manner, and then selects active switches using the same consolidation algorithm as Shrink. (2) *LowerBound:* We have described lower bounds on the network energy use for the FatTree and VL2 topologies in the Appendix A. The lower bounds compute the minimum number of switches that need to be active to support traffic equal to the external bandwidth from a given number of servers.

Topologies: We simulate two datacenter topologies that have no oversubscription: a 3456-server FatTree topology made of 24-port switches (Cisco Nexus 2224P, 80 W, 720 count) [9], and a 1280-server VL2 topology made of 24-port ToR switches (Cisco Nexus 2224P, 80 W, 64 count) [9] and 16-port 10 Gigabit core or aggregation switches (Cisco Catalyst 6500, 480 W, 24 count) [8]. We assume all active servers have equal power use (Acer Altos T350 F2, 130W at 60% utilization [29]).

Results: Figure 9 presents our results. We find that Shrink's consolidation significantly reduces the switch energy use over network-unaware server consolidation (Rand-SN) and gives network energy savings close to the lower bound. For instance, on the FatTree (VL2) topology with 25% of active servers, Shrink's switch energy use is 38% (42%) lower than Rand-SN and only 9% (13%) higher than LowerBound. These improvements reduce overall CDC energy use as well. When 25% of servers are active, Shrink's total energy use is lower than Rand-SN by 9% on FatTree and by 10% on VL2. These results suggest that Shrink's simple network-aware consolidation heuristic is sufficient to achieve most of the energy savings possible by a coordinated selection of active servers and switches.

7 RELATED WORK

To our knowledge, this is the first effort at quantifying energy-performance tradeoff for load-, content-, and network-aware consolidation in CDCs. We discuss prior work on these topics below.

Load-aware consolidation: A long line of work has studied load-aware consolidation using theoretical models and system implementations. A number of issues we address have been addressed previously such as load prediction [21], hardware reliability [17, 19] and load balancing [23]. Some issues such as multiple co-hosted services [6] and datacenter cooling costs [1] are outside the scope of our work. While these efforts focus on stateless systems whose application performance primarily depends on server load, CDCs maintain a large amount of state in the form of cached content. Distinct from prior work, we model the energy-performance tradeoff as being dependent on both server load and cache miss rates. Further, we quantify this tradeoff using a real workload and analyze the relative importance of server load and miss rates in determining this tradeoff. One of our key findings is that a load-aware consolidation can achieve a miss rate close to that of an offline content-aware scheme, which helps it to achieve a good tradeoff between energy and latency.

Content-aware consolidation: A few efforts have studied consolidation for content-based systems. Sierra [31] and SCADS [32] study consolidation for a cluster file system and a key-value store respectively. These systems consider consolidation for backend data stores and focus on maintaining availability of all content despite server shutdown, and optimizing content transfers before shutdown/startup. In contrast, CDC acts as a cache and does not need to ensure availability of all content. Since a CDC often does not have sufficient storage to cache the entire workload, our offline content-aware scheme focuses on minimizing miss rates.

Zhu et al. [38] study dynamically scaling a memory cache of small key-value items residing in front of a database and optimize content transfer during shutdown/startup to reduce load on the database. While their server consolidation only considers time-varying loads, our content-aware consolidation additionally accounts for the variations in the relation between storage and miss rate over time and provisions servers to minimize miss rates over a sequence of time intervals. We also find that content transfers before shutdown are likely to yield a small improvement in miss rates since cache warm up time is a small fraction of the active duration of a server in our workloads.

Network consolidation: Consolidation of switches has been studied for both wide-area and datacenter networks [33, 15, 37, 7, 4]. Whereas these efforts study consolidation of switches in isolation from server consolidation, our approach coordinates server and switch consolidation. Compared to a scheme that does server consolidation in a network-unaware manner, our scheme reduces switch energy use by up to 45%. Moreover, prior work focuses mostly on evaluating traffic engineering metrics such as link utilization [33, 15]. Our work evaluates user-perceived delay for a real application and workload and shows that switch consolidation can be performed with a small delay inflation in CDCs.

Global load balancing: Many papers [18, 24, 12, 26] have shown that geographical load-balancing across data centers can exploit the differences in electricity prices and in renewable energy availability at various locations to reduce energy costs, energy use, or non-renewable energy use. In comparison, our work focus on improving energy-efficiency of a single CDC by the use of consolidation. We believe that global load balancing can complement Shrink in reducing energy use and its cost across datacenters.

8 CONCLUSIONS

In this paper, we analyzed the impact of load-, content- and network-aware consolidation on the energy-performance tradeoff in CDCs. Unlike prior work on load-aware consolidation, we quantified its performance impact while accounting for both an increased server load and miss rates. We showed that for an actual CDC workload, significant energy savings are possible with a small inflation in mean, 95-th percentile and 99-th percentile delay. Next, we designed an ideal content-aware consolidation scheme that provisions servers in an offline manner based on a detailed model of the relation between storage and miss rate. Somewhat surprisingly, our comparison showed that even for workloads with a low skew, the content-aware approach provides only small reduction in miss rates over a load-aware approach. Finally, we considered network-aware consolidation that selects the set of active servers and switches in a coordinated manner. We showed that a simple network-aware heuristic significantly reduces energy use over a network-unaware scheme and provides energy use close to a lower bound. Overall, our work demonstrates that simple consolidation schemes are surprisingly effective in achieving a good energy-performance tradeoff in CDCs.

A NETWORK ENERGY LOWER BOUND

We derive lower bounds on the network energy use for FatTree [3] and VL2 [14] under the constraint that n servers that are active must be able to simultaneously send traffic to clients equal to the external bandwidth E via the set of active switches only.

VL2: Let the energy use of each core, aggregation and ToR switch be PC, PA and PT respectively. Let L be the capacity of links between each pair of core and aggregation switches. If c core switches and a aggregation switches be active, then the maximum number of servers that can be supported is $nmax = (a \times c \times L/E)$ and the total energy use of core and aggregation switches is $etotal = (c \times PC + a \times PA)$. We select the optimal values of a_opt and c_opt (by enumerating all values) such that $etotal$ is minimized under the constraint that $nmax > n$. Assuming each ToR switch connects to k servers, the minimum number of ToR switches needed is $\lceil n/k \rceil$. Thus, a lower bound on the total network energy use is $(\lceil n/k \rceil PT) + (c\_opt \times PC + a\_opt \times PA)$.

FatTree: Switches are identical in a FatTree. So, a lower bound the number of active switches gives a lower bound on network energy use also.

Let $m_1, \cdots m_k$ be the active servers in the k pods so that $m_1 + \cdots + m_k = n$. In a pod with m active servers, at least $2\sqrt{m}$ switches must be active. Thus, a total of $(2(\sqrt{m_1} + \cdots + \sqrt{m_k}))$ pod switches must be active.

Let c be the number of active core switches. We claim that the number of active servers in any pod can at most be c. The reason is that a core switch has only one link to switches in each pod, and hence can receive traffic from only one server sending traffic at its outgoing link capacity to external clients. The values of $m_1, \cdots m_k$ that minimizes the number of active pod switches is given by $m_1 = m_2 = \cdots = m_l = c$, $m_{l+1} = (n \bmod c)$, and $m_{l+2} = m_{l+3} = \cdots m_k = 0$, where $l = \lfloor n/c \rfloor$. Let p be minimum number of active pod switches thus computed. Then, the minimum number of total active switches active is given by $(p + c)$.

Computing the minimum number of switches for all possible values of c $(c \leq n, c \leq k^2/4)$ and taking their minimum gives a lower bound on the number of active switches for this topology.

REFERENCES

[1] F. Ahmad and T. Vijaykumar. Joint optimization of idle and cooling power in data centers while maintaining response time. In *ACM Sigplan Notices*, volume 45, pages 243–256. ACM, 2010.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 63–74. ACM, 2008.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review*. ACM, 2008.

[4] M. Andrews, A. Anta, L. Zhang, and W. Zhao. Routing for Energy Minimization in the Speed Scaling Model. In *INFOCOM*, pages 1–9, 2010.

[5] L. A. Barroso and U. Hölzle. The case for energy-proportional computing. *IEEE computer*, 40(12):33–37, 2007.

[6] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle. Managing energy and server resources in hosting centers. In *ACM SIGOPS Operating Systems Review*. ACM, 2001.

[7] L. Chiaraviglio, M. Mellia, and F. Neri. Minimizing ISP network energy cost: formulation and solutions. *IEEE/ACM Transactions of Networking*, 20(2):463–476, Apr. 2012.

[8] Cisco. Catalyst 6500, 2014. http://www.cisco.com/c/en/us/products/switches/catalyst-6500-series-switches/index.html.

[9] Cisco. Data center switches, 2014. http://www.cisco.com/c/en/us/products/switches/data-center-switches/index.html.

[10] Cisco. Visual Networking Index: Forecast and Methodology, 2015-2020, 2015. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html.

[11] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker. Less pain, most of the gain: Incrementally deployable ICN. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 147–158. ACM, 2013.

[12] P. X. Gao, A. R. Curtis, B. Wong, and S. Keshav. It's not easy being green. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '12, pages 211–222, New York, NY, USA, 2012. ACM.

[13] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: A scalable and flexible data center network. In *ACM SIGCOMM computer communication review*, volume 39, pages 51–62. ACM, 2009.

[14] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: a scalable and flexible data center network. In *ACM SIGCOMM Computer Communication Review*. ACM, 2009.

[15] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. ElasticTree: Saving energy in data center networks. In *NSDI*, pages 17–17, Berkeley, CA, USA, 2010. USENIX Association.

[16] C. E. Hopps. RFC 2992: Analysis of an equal-cost multi-path algorithm. 2000.

[17] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska. Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking*, 2012.

[18] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. Andrew. Greening geographical load balancing. In *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, SIGMETRICS '11, pages 233–244, New York, NY, USA, 2011. ACM.

[19] V. Mathew, R. Sitaraman, and P. Shenoy. Energy-aware load balancing in content delivery networks. In *INFOCOM*, pages 954–962, 2012.

[20] Netgear. GS105 5-port Gigabit Ethernet Switch, 2014. http://www.downloads.netgear.com/files/GDC/GS105/GS105_datasheet_04Sept03.pdf.

[21] H. Nguyen, Z. Shen, X. Gu, S. Subbiah, and J. Wilkes. Agile: Elastic distributed resource scaling for infrastructure-as-a-service. In *ICAC*, pages 69–82, 2013.

[22] Nielsen. Online Video Usage Up 45%, 2011. http://www.nielsen.com/us/en/insights/news/2011/january-2011-online-video-usage-up-45.html.

[23] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath. Load balancing and unbalancing for power and performance in cluster-based systems. In *Workshop on compilers and operating systems for low power.*, 2001.

[24] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs. Cutting the Electric Bill for Internet-Scale Systems. In *ACM SIGCOMM*, Barcelona, Spain, August 2009.

[25] K. Rajamani and C. Lefurgy. On evaluating request-distribution schemes for saving energy in server clusters. In *ISPASS*, pages 111–122. IEEE, 2003.

[26] L. Rao, X. Liu, L. Xie, and W. Liu. Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, 2010.

[27] ServerTech. Switched Rack PDUs, 2014. http://www.servertech.com/products/switched-pdus/.

[28] A. Sharma, A. Venkataramani, and R. Sitaraman. Distributing content simplifies isp traffic engineering. In *SIGMETRICS*, 2013.

[29] SPEC. Standard performance evaluation corporation, 2014. https://www.spec.org/power_ssj2008/results/res2013q3/.

[30] Squid. Squid Cache. http://www.squid-cache.org/.

[31] E. Thereska, A. Donnelly, and D. Narayanan. Sierra: Practical power-proportionality for data center storage. In *Eurosys*, pages 169–182. ACM, 2011.

[32] B. Trushkowsky, P. Bodík, A. Fox, M. J. Franklin, M. I. Jordan, and D. A. Patterson. The SCADS Director: Scaling a distributed storage system under stringent performance requirements. In *FAST*, pages 12–12, Berkeley, CA, USA, 2011. USENIX Association.

[33] N. Vasić, P. Bhurat, D. Novaković, M. Canini, S. Shekhar, and D. Kostić. Identifying and using energy-critical paths. In *CoNEXT*, 2011.

[34] D. Wessels and K. Claffy. RFC 2186, version 2: Internet Cache Protocol (ICP). Technical report, IETF, 1997.

[35] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *OSDI*, 2002.

[36] WTI. Remote console and power management systems, 2014. http://www.wti.com/t-remote-console-and-power-management-for-cisco-routers.aspx.

[37] M. Zhang, C. Yi, B. Liu, and B. Zhang. GreenTE: Power-aware traffic engineering. In *ICNP*, pages 21–30, 2010.

[38] T. Zhu, A. Gandhi, M. Harchol-Balter, and M. A. Kozuch. Saving cash by using less cache. In *HotCloud*, 2012.