Shrink: Quantifying and Leveraging Energy-Performance Tradeoff in Content Datacenters

ABSTRACT

Energy use is a key component of operational costs of *content datacenters (CDCs)*, datacenters that are used for storing and serving content to end-users. CDCs can potentially reduce their energy use via consolidation of servers and switches, but in doing so, they risk inflating end-user response times potentially leading to SLA violations. Our primary contribution is to quantify the tradeoff between energy savings via consolidation and response times in CDCs, and the design and implementation of Shrink, a system that aggressively leverages this tradeoff in order to yield significant savings in energy use in CDCs while affecting user-perceived response times in a controlled manner. 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% compared to network-unaware server consolidation schemes. Our evaluation of Shrink using a workload from a large CDN's datacenter shows that Shrink can reduce energy use by 35% over a scheme provisioned for the peak demand, while increasing the mean, 95th and 99th percentile response times by 8%, 3% and 15% respectively.

1. INTRODUCTION

Our consumption of digital content is increasing rapidly. The majority of the traffic on the Internet today is content, with video content by itself accounting for more than half the total traffic [11, 35]. Not surprisingly, many datacenters and Points-of-Presence (PoPs) are dedicated to storing and serving content to end users. We call these datacenters and PoPs *content datacenters* (or, CDCs).

The increasing energy use of datacenters [20, 42, 23] has motivated a long line of research in *consolidation* schemes that aggregate a datacenter's load on a fraction of components and save energy by turning off unused components [8, 31, 40, 27]. Consolidation exploits the over-provisioning of resources in a datacenter and the diurnal variations in load to potentially reduce datacenter energy use by up to 50% [31].

The practicality of consolidation as an energy-saving tool for a CDC operator depends on its impact on end-user response time. End-user response time is a key metric on which operators are evaluated [43]. Further, a sharp response time inflation may be perceived as service unavailability, thereby causing a service-level agreement (SLA) violation and revenue loss for an operator [4, 32, 26]. Despite a large body of literature on consolidation schemes, quantifying the precise impact on user-perceived response times is not well understood today.

Our position is that a lack of quantitative understanding of energy vs. response time tradeoff is potentially a major roadblock to widespread deployment of consolidation to reduce CDC energy use. To provide the insights needed for an operator to reduce its energy use, we address these key questions: (1) What is the energy-response time tradeoff achieved by simple schemes for server and network consolidation in a CDC that uses well-known schemes for load balancing and network routing? (2) What is the additional energy savings achieved by network-aware server consolidation over network-unaware server consolidation considered previously?

Our primary contribution is to quantify the tradeoff between energy savings via consolidation and response times for a real CDC's workload, and the design and implementation of a system, Shrink, to leverage this tradeoff. Shrink reduces the energy use of servers and switches via consolidation, while enabling operators to achieve the desired response time and hardware reliability. Shrink uses a novel network-aware server consolidation scheme that selects the active servers in a left-to-right order in a topology and increases the potential network energy savings. Shrink does not require changes to datacenter network fabric, depending only on ECMP (equal-cost multipath) for routing, support for which is widely available today. As our work is grounded in implementation, it accounts for several factors affecting response time - increased load on servers and network links, reduced storage and its impact on cache hit rates, non-steady state cache behavior due to on-off transitions, imperfect load balancing among servers - and hence, accurately quantifies the impact of consolidation on end-user response time.

We have built a prototype of Shrink using Squid [15],

a caching proxy, and have deployed it on Amazon EC2 [5] and Emulab [52]. To conduct a realistic evaluation, we have collected and used traces containing more than 2 billion requests generating nearly 200 TB of traffic from a datacenter of the world's largest CDN, Akamai. Our key empirical results are the following.

(1) *Comparison to baseline:* Shrink reduces energy use by 35% over a baseline scheme that provisions resources according to the peak demand while increasing mean, 95-th %-ile and 99-th %-ile latency by 8%, 3% and 15% respectively.

(2) *Comparison to ideal:* Shrink achieves a mean, 95-th %-ile and 99-th %-ile response time that is 15%, 3% and 25% higher than a scheme that characterizes the ideal energy-response time tradeoff while using 5% more energy than it.

(3) *Comparison of network energy use:* When one-fourth of the servers are active, Shrink's network energy use is lower than a network-unaware server consolidation scheme by 37% and 42% on FatTree [3] and VL2 [21] respectively.

Scope of the paper: This paper focuses on energy saving schemes based on server and network consolidation deployable at the scale of a single CDC. The following schemes, although related, are outside the scope of this paper: (1) Adapting global load balancing across CDCs to reduce the total energy use of all CDCs. (2) Making each server or switch power-proportional, e.g., server energy use can be reduced by turning off a fraction of its disks. (3) Optimizing the cost of energy used such as by accounting for time-of-day pricing for electricity; Section 6 discusses electricity costs.

2. CONTENT DATACENTER BACKGROUND

The server to handle a particular end user's request is determined by a *load balancer*. Servers in a CDC have local storage used for caching content. When a request reaches a server, it serves the content from its cache, if the content is available there, or else, it may fetch the content from a *peer* cache in the same CDC. If the requested content is unavailable on the CDC's servers, the request results in a *cache miss*. On a cache miss, content is fetched from remote locations, either from another CDC or from *origin servers*, at a cost of an increased response time, and served to the end user.

A CDC reduces end-user response time by avoiding load hotspots on servers and increasing cache hit rates, both with the help of its load balancer. Load balancer distributes request among servers to avoid load hotspots, which keeps server-load-dependent delays small. Further, it serves a content's requests from a small number of servers, which results in a small number of content replicas and increases overall cache hit rates.

CDCs should reduce their energy use because it can bring significant greenhouse emission reduction as well as cost savings to operators. Energy use is known to be 15-20% of the total cost of ownership of datacenters [20, 42, 23]. A CDC operator usually manages a global network of such CDCs, which could comprise of 100K servers or even more [2]. In a network with 100K servers with each server consuming 100 W, a 20% energy savings translates to 17520 MWh of yearly energy savings, which is equivalent to average annual energy use of 1616 homes in the US [1], and at a cost of 10c/KWh translates to $1.752 M in yearly cost savings.

Topology	Server count	Switches	Switch pow. / Server pow.
1 Gbps server link			
FatTree	3456	Cisco 2224TP (80W, 720 count)	0.17
VL2	2880	Cisco 2224TP (80W, 144 count), Cisco Nexus 5548P (390 W, 24 count)	0.08
10 Gbps server link			
FatTree	3456	Cisco Nexus 5548P (390 W, 360 count)	0.40
VL2	2304	Cisco 6001 (750W, 48 count), Cisco 6004 (2900W, 6 count)	0.23

Figure 1 *Ratio of network to server energy use at typical operating conditions. Switch power use data from Cisco [13]. Server power use of Acer Altos T350 F2 at a load of 30% is 98.5 W [46].*

Datacenters, including CDCs, should reduce their network energy use as it consumes a non-trivial fraction of datacenter energy. As shown in Figure 1, switches in the FatTree [3] and VL2 [21] toplogies, which support full bisection bandwidth, consume 8% and 17% of the server energy use at typical operating conditions in networks with a 1 Gbps server link capacity. In networks with a 10 Gbps server link capacity, which are gaining adoption, this fraction increases to 40% and 23% for FatTree and VL2 respectively.

Despite a large body of research on automated server and switch consolidation schemes showing significant potential for reducing energy use, the impact of consolidation on user-perceived response times is not well understood. For example, a common problem approach in the literature [31, 22] is to focus on dynamically estimating spare resources, implicitly assuming that shutting down the spare resources will have little or no impact on response times. However, in reality, it is rarely the case that one can, say, shut down 25 out of 400 servers with negligible response time impact. The precise impact depends on the workload dynamics and whether the application is constrained by compute, memory, disk, and/or network capacity. In the following section, we present a simple model that sheds light on these issues specifically in the context of CDCs.

3. ENERGY VS. RESPONSE TIME MODEL

We present a simple analytical model in order to quantify the relationship between energy savings and

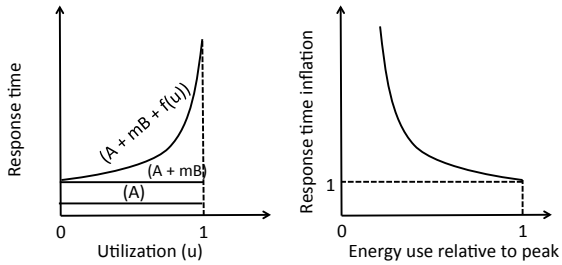


Figure 2 *[Left] An illustrative server utilization vs. response time curve. [Right] Corresponding energy-response time curve.*

response-time inflation in CDCs, and to gain a deeper understanding of the factors that determine this tradeoff. The primary purpose of this model is expository, so we make a number of simplifying assumptions for ease of exposition first, and we progressively relax them in this and subsequent sections.

3.1 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, i.e., the fraction of requests for which the server contacts the origin server. Assume that the power drawn by a server $p(u)$ is a linear function of its utilization u, or the ratio of the incoming load and the server's capacity [31]. In this model, an idle server's power consumption is a fraction I of its peak power P, and the power consumed increases linearly from IP to P as the utilization increases from 0 to 1, i.e., $p(u) = (I + (1 - I)u)P$.

The end-to-end user-perceived response time is assumed to be the sum of three components as also illustrated by the three curves in Figure 2 (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 end-to-end user-perceived delay is $f(u) + mB + A$. As shown in the figure, realistic utilization vs. server delay profile are typically somewhere in between a straight line where the delay increases linearly with the utilization and an L-shaped curve where the delay is zero for all values of utilization less than 1 and is infinity at 1.

3.2 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 power consumption of $(I + (1 - I)U)P$ per server. Thus, the total power consumed is $(I + (1 - I)U)PN$, and the mean response time is $f(U) + mB + A$. 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.

What is the impact of consolidating the datacenter to $n < N$ servers on the total power consumed and user-perceived response time? To answer the first part, we observe that the utilization at each server increases by a factor N/n, so its power consumption would increase to $(I + (1 - I)UN/n)P$. Thus, the consolidated datacenter's total power consumption is given by $(nI + (1 - I)UN)P$, and the corresponding energy use relative to the peak (the x-axis in Figure 2 (right)) is

$$\textit{benefit} = \frac{(nI + (1 - I)UN)P}{(I + (1 - I)U)PN} \tag{1}$$

Computing the corresponding end-to-end response time with the consolidation as above is nontrivial as it requires us to account also for the increase in cache miss rates. Assuming that shutting down servers results in a proportional decrease by a factor N/n in the total available storage (an assumption that is natural for clusters of commodity PCs–the more common option in practice–but not for CDCs relying on network storage), we need to compute the mean miss rate m' that in general would be lower than m. In our numerical examples below, we derive m' using a characteristic-time approximation model for an LRU cache [9]. The response time is given by $f(UN/n) + m'B + A$, and the corresponding response time inflation is given by

$$\textit{cost} = \frac{(f(UN/n) + mB + A)}{(f(U) + m'B + A)}. \tag{2}$$

We make two observations based on the expression for the response time 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 response time 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 response time 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. We numerically exemplify this second insight next.

Numerical example: We evaluate the evaluate the energy-response time tradeoff for Zipfian content popularity distributions. We assume that the server's utilization vs. delay profile is given by $f(u) = Cu^K$, $K >= 1$ is a model parameter and C is a constant delay. Other model parameters are as follows: each server has a capacity = 1 request/sec, total load $L = 15$ requests/sec, latency from clients to servers $A = 10$ ms, latency from servers to origin servers $B = 100$ ms, server delay coef-

ficient $C = 400$ ms, idle power fraction $I = 0.5$, total number of unit-sized content $M = 100$ million, storage per server $S = 1$ million, number of servers $N = 100$.

Figure 3 presents results for workloads with Zipf exponent $\alpha = 1.0$, 0.8 and 0; $\alpha = 0$ results in a uniform distribution. The energy use is shown relative to the peak energy use with $N = 100$ active servers, and response time inflation is shown relative to the least response time in the graph. The server delay model parameter is $K = 9$ for all workloads which implies that the difference in response times among them is due to a difference in miss rates. We observe that workloads with higher Zipf exponents achieve a better energy-response time tradeoff. The reason is that a higher Zipf exponent means that most of the hits result from a small fraction of objects, so reducing the available storage only results in a small increase in miss rates.

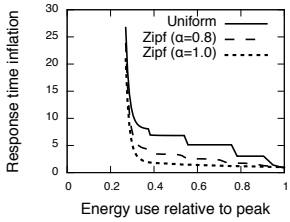


Figure 3 *Energy-response time tradeoff for workloads with varying Zipf exponents.*

Summary: Our simple expository model suggests that the energy vs. response time tradeoff is more favorable as the skew in the server's utilization vs. delay profile increases and the skew in the content popularity distribution increases. Admittedly, this simplistic model has several critical limitations as it (1) does not consider workload dynamics, (2) assumes a perfect load balance among servers, (3) ignores the overhead of coordination between servers, and (4) implicitly assumes that the utilization vs. response time behavior and the cache size vs. miss rate behavior can be approximated by treating the CDC as a single logical server of the same total capacity, ignoring the network fabric entirely.

Thus, a natural question is what do real, achievable energy vs. response time inflation tradeoffs look like in CDCs? Further, does an increase server load or an increase in cache miss rates cause a greater impact on response time? To answer these questions, we present the design and implementation of Shrink and evaluate the tradeoff using a real workload from a large CDC in the next two sections.

4. Shrink DESIGN AND IMPLEMENTATION

Shrink is a system for server and network consolidation as well as load balancing. Shrink reduces a CDC's energy use, while enabling operators to achieve the desired response times and hardware reliability. A novel aspect of Shrink is its network-aware server consolidation scheme, which increases the energy savings that Shrink's network consolidation scheme achieves. This section describes Shrink's design goals, its design and its implementation.

4.1 Design goals

Shrink balances the following three requirements that are important from the perspective of a CDC operator.

(1) **Energy use:** The design should minimize the energy use of a CDC's servers and switches.

(2) **Response time:** The design should enable operators to provide the desired response time to end users.

(3) **Hardware reliability:** The design should enable operators to achieve the desired hardware reliability, where reliability is quantified by the rate of on-off transitions for servers and switches. Note that a decrease in hardware lifetime due to frequent on-off transitions would result in the CDC operator incurring greater capital expenditure over time. For example, if a hard disk rated at 50K start/stop cycles for reliable operation makes one on-off transition per hour, it reaches the start/stop cycles limit in 50K hours, much before its specified mean-time-before-failure of 1.2M hours [44].

4.2 System overview

Shrink runs as a control program that executes at fixed length intervals. In each interval, the control program receives reports from all active servers regarding their load at the granularity of *content buckets*; content are mapped to buckets using a consistent hash function based on their name. Based on three inputs – server load reports, a utilization vs. end-user response time curve for the operator's response time metric of interest (such as a given percentile of response time) and a specified value of end-user response time as per that metric – , the control program computes the following two outputs: (1) **Consolidation:** The set of servers and switches to keep active, while remaining servers and switches are turned off. (2) **Load balancing:** A *traffic split* for each bucket describing the ratio in which the bucket's requests are split among servers. We next describe Shrink's consolidation and load balancing algorithms, followed by how to compute the utilization vs. end-user response time curve provided to Shrink.

4.3 Consolidation

Shrink's consolidation algorithm assumes 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. To adapt network routing in response to network consolidation, Shrink requires the support for ECMP [25], which is commonly available in the datacenter network fabrics today.

4.3.1 Server consolidation

Shrink selects the active servers, in a network-aware manner, to be the leftmost leaf nodes in a topology.

Figure 4 shows why such a network-aware server consolidation can increase the energy savings that network consolidation achieves. In the figure, both the left and the right topologies have 20 active servers. The right topology requires all ToR switches to be kept active, but in the left topology the set of active servers are chosen among servers in one 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 network energy use.

Let $F(u)$ be the utilization vs. end-user response time curve for the operator's response time metric of interest provided to Shrink. $F(u)$ is similar to the end-to-end response time ($f(u) + mB + A$) described in Section 3. We discuss how $F(u)$ can be obtained in Section 4.5. Shrink uses $F(u)$ assuming that if the servers active in the datacenter are run at an average utilization u, then the resulting response time is $F(u)$. Thus, to achieve a specified end-user response time R, Shrink chooses a server utilization limit $U = r^{-1}(R)$.

Shrink computes the number of active servers using two integer values: *minServers* and *maxServers*. *minServers* $= \lceil L/U \rceil$, where the L is the predicted load for the next interval; L is predicted 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; W is called the *pre-shutdown wait interval*. 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.

Reliability of servers: The reliability of servers depends on the pre-shutdown wait interval (W). 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. Our empirical results (Section 5.2.4) show that W close to 1 hr 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× lower than that of $W = 1$ min. Thus, we expect the default value of $W = 1$ hr to work well for operators. However, operators can better inform their choice of W with similar tests for their workload.

4.3.2 Network consolidation

Network consolidation in Shrink satisfies the following condition: assuming ECMP routing, all active servers can simultaneously send traffic to clients equal to the *external bandwidth* using the set of active switches only. We define the external bandwidth as the outgoing link capacity of a server divided by the oversubscription ratio of the datacenter. For example, in a network with 2:1 oversubscription and 1 Gbps server outgoing links,

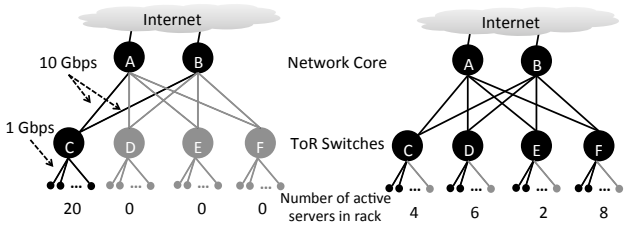


Figure 4 *Black (grey) components are turned on (off). Each rack has 20 servers. (Right) Randomly choosing the set of active servers results in all ToR switches being turned on. (Left) Choosing the same number of active servers in a "network-aware" manner enables more ToR switches to be turned off.*

the external bandwidth is 500 Mbps. If a server sends traffic at a maximum rate up to the external bandwidth, then this condition avoids network bottlenecks.

We explain the above observation using an example. Consider the topology in Figure 4 (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 switch C to core switches A & B is (800 Mbps)×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 link AC.

Shrink's network consolidation assumes that each active server is sending traffic to clients at a rate equal to the external bandwidth. It selects the set of switches needed to route this traffic to the root nodes. It considers 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. Each switches selects p-leftmost parents that must be active to forward the traffic sent by its children to 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. Due to ECMP support, traffic forwarded to root nodes is divided on links to the p parent nodes equally. Finally, only the switches that are transmitting non-zero traffic are selected to be active.

Reliability of switches: Network consolidation in Shrink satisfies the following property related to reliability of switches: 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. Due to this property, the rate of on-off transitions of switches is close to that of on-off transitions for servers. 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

```
   input  : buckets // set of content buckets
          : bktsLoad // content bucket to load map
          : srvrs // set of active servers
          : U // server utilization limit
          : k // default server count serving a content bucket
          : trSplits // bucket to traffic split map (old)
   output: newTrSplits // bucket to traffic split map (new)
1  limit= max(∑_{bkt∈buckets} bktsLoad[bkt] / |srvrs|, U) // server load limit
2  srvrsLoad = {} // server to load map
3  for srvr∈ srvrs do
4  |  srvrsLoad[srvr] = 0
5  newTrSplitBkts = {} // buckets needing new traffic splits
6  for bkt ∈ buckets do
7  |  for (s_j^bkt, f_j^bkt) ∈ trSplits[bkt] do
8  |  |  // server s_j^bkt serves a fraction f_j^bkt of bkt's load
9  |  |  if s_j^bkt ∉ srvrs or
   |  |  srvrsLoad[s_j^bkt] + f_j^bkt bktsLoad[bkt] > limit then
10 |  |  |  Add bkt to newTrSplitBkts
11 |  |  |  break
12 |  if bkt ∉ newTrSplitBkts then
13 |  |  for (s_j^bkt, f_j^bkt) ∈ trSplits[bkt] do
14 |  |  |  srvrsLoad[s_j^bkt] += f_j^bkt bktsLoad[bkt]
15 |  |  trSplits[bkt] = newTrSplits[bkt]
16 for  bkt ∈ newTrSplitBkts do
17 |  select k servers (s_1, s_2, ...s_k) randomly from srvrs
18 |  bktTrSplit= {} // new traffic split for bkt
19 |  remBktLoad = bktsLoad[bkt] // remaining load for bkt
20 |  for srvr ∈ (s_1, ...s_k) do
21 |  |  if srvrsLoad[srvr] + bktsLoad[bkt]/k < limit then
22 |  |  |  bktTrSplit = bktTrSplit ∪ {(srvr, 1/k)}
23 |  |  |  remBktLoad −= bktsLoad[bkt]/k
24 |  while remBktLoad > 0 do
25 |  |  select minLoadSrvr whose srvrsLoad[minLoadSrvr]
   |  |  is the least
26 |  |  load = min(remBktLoad, limit −
   |  |  srvrsLoad[minLoadSrvr])
27 |  |  remBktLoad −= load
28 |  |  Add {(minLoadSrvr, load / bktsLoad[bkt])} to bktTrSplit
29 |  newTrSplits[bkt] = bktTrSplit
```

Algorithm 1: Load balancing

transition from off to on state for each server. Due to the above property, the set of active switches can only add new switches as the number of active servers increases. As a result, each switch would also have at most one transition from off to on state in this interval. Thus, the above property ensures that improving server reliability improves switch reliability as a consequence.

4.4 Load balancing

Shrink uses randomized load balancing over a set of content buckets. Content is mapped to a fixed number of buckets (default = 100) using a consistent hash function based on its name. The output of load balancing is a *traffic split* for each bucket that determines the ratios in which the bucket's requests will be distributed among CDC's servers. Traffic split for a bucket is determined based on the load predicted for the bucket in the next interval. Shrink predicts a bucket's load using a linear regression model trained based on the observed load for the bucket in the past few intervals.

Shrink's load balancing (Algorithm 1) selects a load limit for servers, which is greater of the server utilization limit U or the average load on an active server (line 1). The algorithm selects buckets that need new traffic splits compared to the previous execution of the algorithm. These buckets either belong to servers that are no longer active or whose load exceeds the load limit (lines 5-15). To compute new traffic splits for the selected buckets, Shrink selects a fixed number of randomly chosen servers (default = 2) and divides the load for the bucket among them equally. In doing so, if the load on any server exceeds the load limit then Shrink assigns the excess load for that bucket from that server to the least loaded server that is active (lines 16-29).

Load balancing mitigates disruptions to the existing connections by not sending requests to a server that is likely to be turned off. To this end, Shrink's node selection module informs the load balancing module that a server is likely to be turned off in the middle of the pre-shutdown wait interval of the server. As the pre-shutdown wait interval is tens of minutes long, existing connections, except for the long transfers, complete before server shutdown. To mask the server turnoff for these few long transfers, CDCs can use techniques such as IP address takeover [17] or connection migration across machines [45, 47]. We have not implemented these techniques in Shrink.

4.5 Computing utilization vs. response time curve

We discuss three ways of computing the utilization vs. response time curve $F(u)$ input to Shrink in the order of increasing complexity. The more complex schemes potentially enable Shrink to provide response times that are closer to those specified by operators.

The first and second approaches require prior measurements of a single server and of the entire CDC respectively. These measurements obtain the value of the operator's response time metric of interest at varying levels of utilization. These measurements are done with a representative workload in a test environment with real or emulated client-to-server delay and server-to-origin delay. The first approach implicitly assumes that if the response time of a single server at utilization u is r, then the response time of the CDC whose servers have an average utilization u is also r. This assumption ignores the inefficiencies of a distributed system such as imperfect load balancing and coordination overhead among peer caches in a CDC. These factors are more accurately accounted for in the second approach. Nonetheless, both approaches could be inaccurate as they do not consider the non-steady cache behavior due to on-off transitions and increased cache miss rates due to reduced storage. To account for these inaccuracies, a constant inefficiency factor ρ is added as follows. If $F'(u)$ is the measured utilization vs. response time

curve, then Shrink is input a curve $F(u) = \rho \times F'(u)$.

The third approach is to equip Shrink to learn the curve automatically based on online measurements. The overhead of these measurements can be kept small by doing them for a small fraction of requests only. This approach could be more accurate than the previous two approaches because it considers the factors discussed above and can even adapt to changes in workload characteristics that influence the utilization vs. response time behavior. We have not implemented this third approach in Shrink.

4.6 Implementation

Our Shrink prototype is implemented in nearly 6K lines of Java code. By default, Shrink's control program runs at 20 sec intervals. Shrink uses Squid as a caching proxy [15]. We configured Squid to use its AUFS storage, which handles disk accesses asynchronously by multiple threads; AUFS results in lower response over synchronous disk access storage mechanisms. We have added support for content chunking in order to reduce origin traffic and improve cache hit rates: specifically, a large file is requested as a sequence of 2 MB chunks, which enables Squid to cache each chunk independently.

Load balancing: Shrink resembles a DNS load balancer. Similar to a DNS resolution, a client contacts Shrink to receive the traffic split for a content, and then the client contacts a server as per the traffic split to download the content. An alternative would have been to implement Shrink as a middlebox. As a middlebox routes the traffic between clients and servers through it, it would have become a network bottleneck on our testbed. In comparison, a DNS-type load balancer does not become a network bottleneck as it only exchanges load balancing queries and responses from clients.

Server load measurement: We measure a server's load in terms of request rates. We use an approach that requires no modification to the Squid codebase. We run a process on each server that parses the content access logs output by Squid, and sends to Shrink, once every interval, the number of requests for each content bucket at that server in the previous interval.

Peer caching overhead: Each Squid instance runs as an independent cache that fetches content from origin servers upon a cache miss. We tested a configuration of Squid in which all servers were configured as cache peers that used Internet Cache Protocol (ICP) [51] for querying peer caches upon a cache miss. But, we found the overhead of ICP to be nearly 25% of the overall request load while the peer cache hits were less than 3%. Hence, we disabled ICP due to its lower benefit compared to its overhead.

A limitation of our prototype is the lack of 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. Finally, we note that remote power management that is necessary for turning servers and switches on and off is available from multiple vendors today [14, 36].

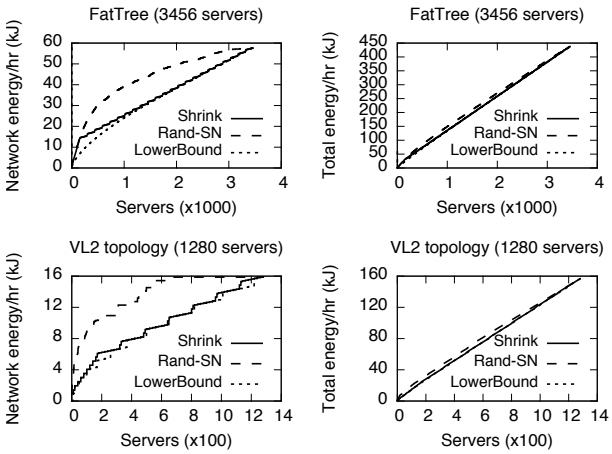


Figure 5 *[Numerical computation] Shrink's network energy use is lower than a network-unaware server consolidation scheme Rand-SN by 38% on FatTree and 42% on VL2 when one-fourth of the servers are active in each topology.*

5. EXPERIMENTAL EVALUATION

Our evaluation has two main goals: (1) Comparing the network energy use of Shrink against a network-unaware server consolidation scheme (Section 5.1). (2) Quantifying the energy-response time tradeoff achieved by Shrink and compare it to the ideal energy response time tradeoff achievable (Section 5.2).

5.1 Comparing network energy use

Schemes compared: (1) *Rand-SN:* Rand-SN selects the set of active servers randomly; it uses the same network consolidation scheme as Shrink. Rand-SN is used to evaluate the benefit of network-aware server consolidation in Shrink. (2) *LowerBound:* We define lower bounds on the network energy use for a given number of active servers on the FatTree and the VL2 topologies. Our computation of LowerBound for FatTree and VL2 is described in Appendix A. Rand-SN and LowerBound provide the same per-server bandwidth guarantee to external hosts as Shrink does.

Topologies: We simulate two network topologies: a 3456-server FatTree topology made of 24-port switches (Cisco Nexus 2224P, 80 Watt, 720 count) [13] consuming 80 Watt per switch, and a 1280-server VL2 topology made of 24-port ToR switches (Cisco Nexus 2224P, 80 W, 64 count) [13] and 16-port 10 Gigabit core or aggregation switches (Cisco Catalyst 6500, 480 W, 24 count) [12]. We assume all active servers have identical power use (Acer Altos T350 F2, 130W at 60% utilization [46]).

Results: Figure 5 presents our results. The relative difference between Shrink and Rand-SN reduces

as the number of active servers increases. When 25% and 50% of servers are active, Shrink's network energy use is lower than Rand-SN by 38% and 26% respectively on FatTree and 42% and 35% respectively on VL2. When 25% and 50% of servers are active, Shrink's network energy use higher than LowerBound by 9% and 2% respectively on FatTree and by 13% and 7% respectively on VL2. These findings show that Shrink's network-aware server consolidation reduces the network energy use over network-unaware server consolidation schemes and gives network energy savings close to the lower bound. When 25% and 50% of servers are active, Shrink's *total* energy use is lower than Rand-SN by 9% and 5% respectively on FatTree and 10% and 7% respectively on VL2. Thus, Shrink's network-aware server consolidation is effective in reducing aggregate CDC energy use as well.

5.2 Quantifying energy-response time tradeoff

5.2.1 Experiment setup

Akamai dataset: Our evaluation uses content access traces from an Akamai datacenter. The traces include all requests received at a datacenter with 24 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 network traffic.

Testbeds: We use prototype-based experiments (on EC2 and Emulab) and trace-based experiments. Our experiment on EC2 evaluates the energy-response time tradeoff due to server-only consolidation. As we do not have control over network topology on EC2, we use Emulab to evaluate the response time inflation due to both server and network consolidation. Finally, we conduct larger-scale trace-based experiments on a simulator.

Schemes compared: We compare Shrink against *Peak-S* and *Single-server-ideal*. Peak-S represents a baseline in which a CDC operator does not use consolidation to reduce energy use, i.e., it keeps all servers and switches active.

Single-server-ideal is a computation of the ideal energy-response time curve, unachievable by any real system. The points on this curve are obtained by varying the utilization u up to which any server can be loaded. For a given u, the response time of Single-server-ideal for a given metric is equal to the measured load-vs.-response time curve of a single server for the same metric at the same utilization. For the same utilization u, any distributed system will have a higher response time because workload dynamics, load imbalance and non-steady state cache behavior; these factors are ignored by Single-server-ideal. Our single server measurements are done with a representative workload in the respective experimental environment, EC2 or Emulab, with emulated client-to-server delay and server-to-origin delay.

For Single-server-ideal, we compute the set of servers and switches in each time interval that minimizes the total energy use as follows. Based on four inputs – u, the total load in each time interval, the number of server transitions allowed, and the power model of each server –, we use a dynamic programming algorithm to compute the total energy use of servers and the number of active servers in each time interval. The number of transitions is equal to one on-off server transition/server/day or the same number of transitions as Shrink in that experiment, whichever is higher. The ideal network energy use for the tree topology we experiment with (Section 5.2.3) is computed based on the number of active servers in each time interval. The set of active servers are selected in a left-to-right order; for each active server, we select the switches on the path to the root. The set of switches selected across all active servers is the set that optimizes network energy use.

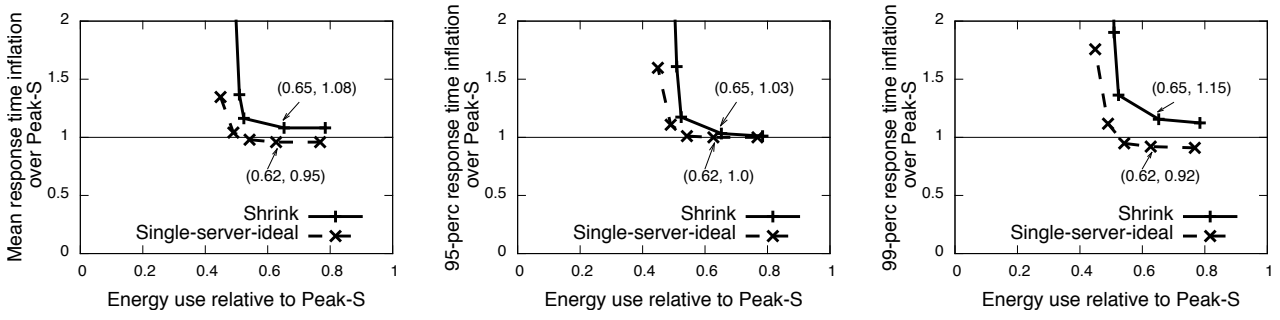
5.2.2 Prototype-based experiments: server consolidation

This experiment quantifies the energy-response time tradeoff due to server consolidation on EC2. Our EC2 testbed consists of 15 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. We use a short pre-shutdown wait interval $W = 10$ min for Shrink because our workload is a sped up by 8×.

We calculate the energy savings relative to Peak-S as per Equation 1; the ratio of the idle to peak energy use of servers, I equals 0.5 [7]. Peak-S uses 15 servers in this experiment. We have conservatively chosen the number of servers in Peak-S to be much less than the number of servers in the Akamai datacenter itself so as not to overestimate the energy savings.

We evaluate Shrink in terms of three response time



(a) Mean *(b) 95-th percentile* *(c) 99-th percentile*

Figure 6 *Server consolidation (Section 5.2.2): Shrink's reduces energy over Peak-S with a small response time inflation. In Figure 6a, Shrink's energy use is 0.65× of Peak-S and its mean response time is 1.08× of Peak-S; Single-server-ideal's energy use is 0.62× of Peak-S and its mean response time is 0.95× of Peak-S.*

metrics – mean, 95-percentile, 99-percentile. To provide a utilization-vs.-response time curve $F(.)$ to Shrink for each metric, we take the first approach discussed in Section 4.5. The function $F(.)$ for each metric is equal to the measured utilization vs. response time curve of a single server with an inefficiency factor $\rho = 0.2$. Based on $F(.)$ for each metric, we specify response times $F(u)$ to Shrink for values of u from 0.375 to 0.875 at intervals of 0.125 across different runs.

Figure 6 compares the response time and the energy use of Shrink relative to Peak-S for the mean, the 95-th percentile and the 99-th percentile of response times. 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 7 (left) 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. This result implies that an operator for which these inflations are tolerable, e.g., they do not cause an SLA violation, can achieve the corresponding energy savings as well.

Does an increase server load or a decrease in cache hit rates cause a greater impact on Shrink's response time over Peak-S? In Figure 7 (right) the x-axis shows the server utilization limit U computed by Shrink's consolidation algorithm (Section 4.3.1) and y-axes show corresponding the hit rates and the mean response time of Shrink. Shrink's hit rates are lower than Peak-S but the decrease is less than 7% across all utilizations. Thus, the response time inflation due to a decrease in hit rates is likely to be small. 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 3 has suggested that consolidation reduces hit rates by a small fraction for for real workloads with a high skew in content popularity. We find that mean response times 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, there is a small response time inflation due to a decrease in hit rates but severe inflation occurs at high server utilization limits, and is likely due to an increase in server load.

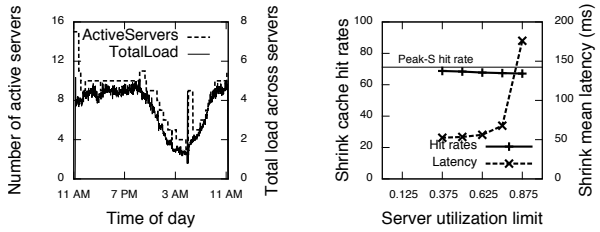


Figure 7 *Server consolidation (Section 5.2.2): [Left] Shrink adapts the number of active servers based on CDC load to reduce energy over Peak-S. [Right] Cache hit rates and mean response time for Shrink and Peak-S.*

Comparing Shrink with Single-server-ideal, in Figure 6a, Shrink's energy use is 0.65× of Peak-S and its mean response time is 1.08× of Peak-S; Single-server-ideal's energy use is 0.62× of Peak-S and its mean response time is 0.95× of Peak-S. There are two reasons that explain the gap between Shrink and Single-server-ideal. First, Single-server-ideal ignores several factors that increase response time of any distributed system such as workload dynamics, load imbalance and non-steady state cache behavior. Second, Shrink waits for the pre-shutdown wait interval to see if a decrease in load persists before turning servers off. But, in our calculation, Single-server-ideal knows the load for the entire experiment beforehand and hence it can shutdown servers sooner than Shrink and save more energy.

5.2.3 Prototype-based experiments: server & network consolidation

We use Emulab to evaluate the energy-response time tradeoff when both server and network consolidation are being performed. For our experiment, we configure a tree topology with 1 Gbps links as shown in Figure 8

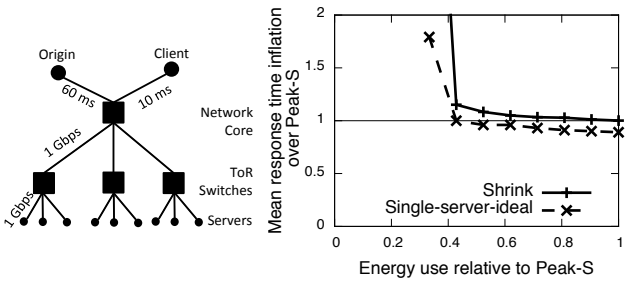


Figure 8 *Network and server consolidation (Section 5.2.3): [Left] Emulab topology for the experiment. [Right] Compared to Peak-S, Shrink has a 15% higher response time but a 57% lower energy use.*

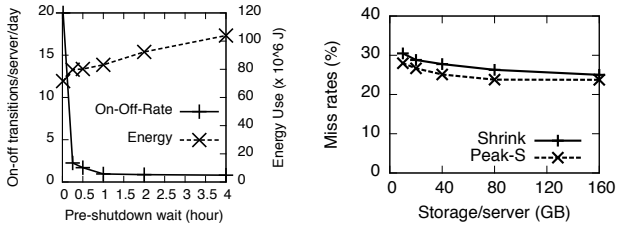


Figure 9 *[Simulator] [Left] Pre-shutdown wait interval between 30 min & 1 hour keeps on-off transition rate close to 1/server/day. [Right] Comparison of miss rates for varying amounts of storage.*

(left). In this topology, the ToR switches have 4 ports, and the core switch has at least 4 ports. Accordingly, we calculate energy use of switches based on the power use of 4-port switch (Netgear GS105), which is 14.4 W [34]. The energy use of servers is computed using the same function as in the previous experiment. Our workload consists of a 1-hour duration of the trace containing requests for one-eighth of the content selected randomly.

Figure 8 (right) compares schemes in terms of the mean response times. Across different runs that vary specified mean response times, 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 three times higher than the peak link utilization for Peak-S. Despite this increase, Shrink's response time is only 15% higher than Peak-S, while its network energy use is 50% lower and the overall energy use is 57% lower than Peak-S (second point from the left in Figure 8 (right)). This result shows that both network and server consolidation can be performed with a small performance impact in CDCs. Finally, we note that the difference between Single-server-ideal and Shrink is consistent with the difference between them in our experiment with server-only consolidation,, e.g., for the same energy savings as Shrink (= 57%), Shrink's response time is 15% more than Single-server-ideal.

5.2.4 Trace-based experiments

Methodology: We conduct experiments for a CDC with 16 servers for the week-long Akamai trace. The capacity of each server is defined in terms of network traffic it can support. The rate of network traffic generated by a request is a constant equal to the client bandwidth reported in the Akamai trace. 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 capacity of each server to be 150 Mbps, and the cache size per server to be 150 GB. Since trace-based experiments do not provide an accurate estimate of response times, we used a fixed server utilization limit $U = 0.65$ for our experiments, which is expected to cause a small response time inflation (Figure 7 (right)). The cache hit rates of our simulator's LRU caching and Squid differ by less than 2% for the same workload and cache size.

Impact on hardware reliability: Figure 9 (left) shows the rate of on-off transitions per server and the corresponding energy savings is achievable. We find that a short pre-shutdown wait interval $W = 1$ min hurts hardware reliability by increasing the rate of server transitions to more than 20/server/day. On the other hand, a high $W = 4$ hours reduces server transition rate to 0.81/server/day, but increases energy use by nearly 45% over $W = 1$ min. The sweet spot for pre-shutdown wait interval is between 30 min to 1 hour, where increase in energy use over $W = 1$ min is between 12% and 15% but most of the reduction in on-off transition rates can still be achieved.

Storage vs. cache miss rates: To determine the sensitivity of miss rates to available storage, we evaluate Peak-S and Shrink for varying amount of storage from 10 GB/server to 160 GB/server and present results in Figure 9 (right). We remark that we have scaled down the CDN trace and hence the storage by a 8× factor, i.e., we would have provisioned 1.28 TB storage instead of 160 GB if we were to experiment with the full trace. As storage reduces, the miss rates increase as expected. But, the relative difference between miss rates for both Shrink and Peak-S remains nearly the same even on reducing the storage to 10 GB, e.g., Shrink's miss rates are 10.3% higher than that of Peak-S for 160 GB storage and are 9.6% higher than that of Peak-S for 10 GB storage. Thus, we conclude that server consolidation schemes are expected to increase datacenter miss rates by a small fraction within the range of storage typically available on server-class machines.

Cooperative caching benefit: We evaluate the potential benefit of cooperative caching among servers in a CDC assuming that a cache-coordination protocol with a much smaller overhead can be developed in future. The hit rates at cache peers are 1.65% for Peak-S and 3.02% for Shrink, which suggests that cooperative

caching among datacenter servers, if implemented efficiently, could reduce the impact of energy optimization schemes by a small margin.

6. DISCUSSION

Energy use vs. energy cost: There are three types of CDCs in terms of their energy cost to an operator.

(1) Operator-owned facility: If a CDC operator owns the datacenter facility, it directly pays to the electricity companies based on its usage. In such datacenters, a reduction in energy use by Shrink is likely to bring a reduction in electricity costs as well.

(2) Co-location facility: A CDC at a co-location facility typically pays by the provisioned power and not the electricity used [39]. Therefore, a reduction in energy use will not bring cost savings to CDC operator with the existing pricing models. However, it is possible a CDC operator may use the reduced energy as a leverage for negotiating a cheaper pricing.

(3) Co-location inside ISP networks: A CDC at a co-location facility maintained by an ISP often has a symbiotic relation with the ISP, where the CDC caches content to reduce the inter-domain traffic for the ISP while an ISP provides co-location free of charge [19]. In such CDCs, energy savings do not translate to cost savings to the CDC operator. Although, energy savings do benefit the ISP, who eventually pays for the electricity.

The type of usage-based energy pricing also determines the cost savings for an operator. Specifically, we distinguish between flat rate pricing and time-of-use pricing [37]. With a flat rate pricing, a given percentage reduction in the energy use results in the same percentage reduction in the energy cost. With a time-of-use pricing, the percentage reduction in the energy use and the energy cost may not be the same. For example, if the peak load on a CDC coincides with the peak hour of electricity prices, the percentage reduction in the energy cost would be lower than the percentage reduction in the energy use.

Impact on web-page load time: Our prototype-based experiments evaluate the response time for individual HTTP requests, and hence do not capture a key metric that is more relevant from an end-user's perspective: web-page load time. However, we expect the inflation in web-page load time to be lower than the inflation in response times given that computation in web browsers constitutes up to 35% of the critical path of a web-page load time [50].

7. RELATED WORK

Our effort distinguishes from prior work in quantifying the energy-response time tradeoff in CDCs, presenting the design and implementation of a system to leverage this tradeoff and proposing a network-aware server consolidation scheme to reduce network energy use. Prior work on reducing energy of datacenters can be divided into three topics: (1) power-proportionality of servers and switches. (2) server and network consolidation in a datacenter and (3) global load balancing across datacenters.

Power-proportional servers and switches: Several efforts have focused on reducing energy use of a server's sub-systems such as CPU [53], disk [29], and memory [16]. Similarly, Nedevschi et al. [33] study power management for switches that support sleep states or several power/performance states similar to CPUs. Nonetheless, today's servers and switches are far from power-proportional. Mahadevan et al. show that networking equipment consumes 62%-91% of their peak energy in idle state [30] and servers consume 32% to 42% of maximum power at a small utilization of 10% [46]. Until the ideal of power-proportionality is achieved, consolidation remains a promising approach to save energy.

Server consolidation: Given the long line of work in server consolidation, our work does not focus on saving more energy than the existing consolidation schemes, but instead on accurately quantifying the impact on response times of a simple consolidation scheme.

The analytical work in this area shares similar goals as us. Lin et al. [27] propose an algorithm for optimizing a cost metric that incorporates energy costs, on-off switching costs and cost of degradation in performance. Mathew et al. [31] propose an algorithm that balances energy use, reliability and availability of servers, which they evaluate based on load traces from Akamai datacenters. In comparison, our implementation-based approach enables us to accurately model the relation between server utilization and response time, impact of server consolidation on cache hit rates, and non-ideal load balancing, to accurately quantify the impact of consolidation on response time for CDCs.

Several efforts have conducted an implementation-based evaluation of server consolidation for stateless systems. Chase et al. [8] allocate resources among multiple co-hosted services in a cluster while reducing energy via consolidation. Pinheiro's [38] system proposes consolidation and load balancing algorithms given a bound on the performance degradation that is acceptable. Rajamani et al. [40] evaluate consolidation schemes for a modified TPC-W workload. In comparison, our effort focuses on CDCs that maintain a large amount of state in the form of cached content. In CDCs, the effect of consolidation on response times cannot be evaluated accurately without accounting for the effect of consolidation on the availability of cached content and the resulting cache hit rates.

Trushkowsky et al. [48] dynamically allocate servers and reconfigure the data stored on the servers to meet service-level objectives such as 99-th percentile request latency. However, there are two key differences between

their work and ours. First, they focus on a workload exclusively of small (256B) objects stored in-memory, whereas CDCs need to deal with orders of magnitude of heterogeneity in object sizes and extensively use a disk cache to improve hit rates. Second, their system appears to be a backend data store, which always has content available within the datacenter. In comparison, CDCs have a significant fraction of traffic to remote datacenters due to cache misses, and the impact of consolidation on response time in CDCs depends on the increase in traffic to remote datacenters that consolidation causes. For these reasons, it is not clear if their findings on the impact of dynamic server allocation on request latency would be applicable for CDCs.

Network consolidation: Network consolidation has been studied for both wide-area and data center networks [49, 24, 54, 10, 6]. Network consolidation concentrates traffic, represented in the form of a traffic matrix, on a subset of links and switches, and turns off remaining switches and links to save energy. Our work differentiates from prior work in two ways. First, prior work evaluates schemes mostly using traffic engineering metrics such as link utilization, while we evaluate actual end user response time for a real application and show that network consolidation can be performed with a small performance impact in CDCs. Second, we show network consolidation is closely related to server consolidation. Our network-aware server consolidation saves up to 45% more network energy over a network-unaware server consolidation scheme.

Global load balancing: Many papers [28, 39, 18, 41] 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

Despite much prior work on reducing the energy use of datacenters via server and network consolidation, a major barrier to their widespread adoption today is that operators worry deeply about the impact on SLAs or user-perceived response times, but the precise relationship between energy savings and response times is not well understood. Our work takes a step towards addressing this concern by presenting a model to quantify the energy savings vs. response time inflation trade-off in CDCs. Based on these insights, we have designed and implemented Shrink, a system that leverages this trade-off to yield significant energy savings while affecting user-perceived response times in a controlled manner. Shrink's novel network-aware server consolidation algorithm reduces network energy use by up to 42% compared to network-unaware server consolidation schemes. Our experiments based on content access traces from an Akamai datacenter showed that our energy optimization algorithms reduce energy use by 35% compared to a conservative baseline scheme that provisions servers as per the peak demand while increasing mean response time by 8% over it. Overall, our findings encourage deployment of energy optimization techniques in CDCs.

APPENDIX

A. NETWORK ENERGY LOWER BOUND

We derive lower bounds on the network energy use for FatTree [3] and VL2 [21] 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.

B. REFERENCES

[1] U. E. I. Administration. FAQs. 2014. http://www.eia.gov/tools/faqs/.

[2] Akamai. Facts and figures, 2014. http://www.akamai.com/html/about/facts_figures.html.

[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] Amazon. Cloudfront service level agreement, 2013. http://aws.amazon.com/cloudfront/sla/.

[5] Amazon. Ec2, 2014. http://aws.amazon.com/ec2/.

[6] M. Andrews, A. Anta, L. Zhang, and W. Zhao. Routing for energy minimization in the speed scaling model. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, 2010.

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

[8] 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.

[9] H. Che, Y. Tung, and Z. Wang. Hierarchical web caching systems: Modeling, design and experimental results. *Selected Areas in Communications, IEEE Journal on*, 20(7):1305–1314, 2002.

[10] L. Chiaraviglio, M. Mellia, and F. Neri. Minimizing isp network energy cost: formulation and solutions. *IEEE/ACM Trans. Netw.*, 20(2):463–476, Apr. 2012.

[11] Cisco. Visual Networking Index: Forecast and Methodology, 2013■2018, 2013.

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

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

[14] R. console and power management systems. Wti, 2014. http://www.wti.com/t-remote-console-and-power-management-for-cisco-routers.aspx.

[15] S. O. W. Delivery. Squid: Optimising Web Delivery. http://www.squid-cache.org/.

[16] X. Fan, C. Ellis, and A. Lebeck. Memory controller policies for dram power management. In *Proceedings of the 2001 international symposium on Low power electronics and design*, pages 129–134. ACM, 2001.

[17] R. Flickenger. *Linux server hacks: 100 industrial-strength tips and tools*, volume 1. " O'Reilly Media, Inc.", 2003.

[18] 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.

[19] Google. Google caching overview, 2014. https://peering.google.com/about/ggc.html.

[20] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel. The cost of a cloud: Research problems in data center networks. *SIGCOMM Comput. Commun. Rev.*, 39(1):68–73, Dec. 2008.

[21] 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.

[22] B. Guenter, N. Jain, and C. Williams. Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning. In *INFOCOM, 2011 Proceedings IEEE*, pages 1332–1340, 2011.

[23] J. Hamilton. Cost of power in large-scale data centers, 2008. http://perspectives.mvdirona.com/2008/11/28/CostOfPowerInLargeScaleDataCenters.aspx.

[24] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. Elastictree: saving energy in data center networks. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10, pages 17–17, Berkeley, CA, USA, 2010. USENIX Association.

[25] C. E. Hopps. Analysis of an equal-cost multi-path algorithm. 2000.

[26] HP. Service level agreement for hp cloud cdn, 2014.

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

[28] 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.

[29] Y.-H. Lu and G. De Micheli. Adaptive hard disk power management on personal computers. In *Great Lakes Symposium on VLSI*, pages 50–50. IEEE Computer Society, 1999.

[30] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan. A power benchmarking framework for network devices. In *NETWORKING 2009*, pages 795–808. Springer,

2009.

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

[32] Microsoft. Windows azure support, 2014.

[33] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 323–336, Berkeley, CA, USA, 2008. USENIX Association.

[34] Netgear. Gs105 5-port gigabit ethernet switch, 2014. `http://www.downloads.netgear.com/files/GDC/GS105/GS105_datasheet_04Sept03.pdf`.

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

[36] S. PDU. Server tech, 2014. `http://www.servertech.com/products/switched-pdus/`.

[37] PG&E. Rate information center, 2014. `http://www.pge.com/en/mybusiness/rates/rateinfo/index.page`.

[38] 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.

[39] 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.

[40] K. Rajamani and C. Lefurgy. On evaluating request-distribution schemes for saving energy in server clusters. In *Performance Analysis of Systems and Software, 2003. ISPASS. 2003 IEEE International Symposium on*, pages 111–122. IEEE, 2003.

[41] 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.

[42] N. Rasmussen. Determining total cost of ownership for data center and network room infrastructure. *White Paper*, 2011.

[43] D. Rayburn. Comparing CDN Performance: Amazon CloudFront Last Mile Testing Results. 2012.

[44] Seagate. Barracuda es.2 serial ata product manual, 2009. `http://www.seagate.com/staticfiles/support/disc/manuals/NL35%20Series%20&%20BC%20ES%20Series/Barracuda%20ES.2%20Series/100468393h.pdf`.

[45] A. C. Snoeren, D. G. Andersen, and H. Balakrishnan. Fine-grained failover using connection migration. In *USITS*, volume 1, pages 19–19, 2001.

[46] SPEC. Standard performance evaluation corporation, 2014.

[47] F. Sultan, K. Srinivasan, D. Iyer, and L. Iftode. Migratory tcp: Connection migration for service continuity in the internet. In *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on*, pages 469–470. IEEE, 2002.

[48] 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 *Proceedings of the 9th USENIX Conference on File and Stroage Technologies*, FAST'11, pages 12–12, Berkeley, CA, USA, 2011. USENIX Association.

[49] N. Vasić, P. Bhurat, D. Novaković, M. Canini, S. Shekhar, and D. Kostić. Identifying and using energy-critical paths. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, CoNEXT '11, pages 18:1–18:12, New York, NY, USA, 2011. ACM.

[50] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall. Demystifying page load performance with wprof. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, 2013.

[51] D. Wessels and K. Claffy. Rfc 2186: Internet cache protocol (icp). Technical report, version 2. RFC, IETF, 1997.

[52] 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.

[53] Wikipedia. Voltage and frequency scaling, 2014. `http://en.wikipedia.org/wiki/Voltage_and_frequency_scaling`.

[54] M. Zhang, C. Yi, B. Liu, and B. Zhang. Greente: Power-aware traffic engineering. In *Network Protocols (ICNP), 2010 18th IEEE International Conference on*, pages 21–30, 2010.