Energy-Efficient Content Delivery Networks using Cluster Shutdown

Vimal Mathew[a], Ramesh K. Sitaraman[a,b], Prashant Shenoy[a]

[a] *University of Massachusetts, Amherst*
[b] *Akamai Technologies Inc.*

Abstract

Content delivery networks (CDNs) are an important class of Internet-scale distributed systems that deliver web, streaming, and application content to end users. A commercial CDN could comprise hundreds of thousands of servers deployed in over thousand clusters across the globe and incurs significant energy costs for powering and cooling their servers. Since energy costs are a significant component of the total operating expense of a CDN, we propose and explore a novel technique called cluster shutdown that turns off an entire cluster of servers of a CDN that is deployed within a data center. By doing so, cluster shutdown saves not just the power consumed by the servers but also the power needed for cooling those servers. We present an algorithm for cluster shutdown that is based on realistic power models for servers and cooling equipment and can be implemented as a part of the global load balancer of a CDN. We evaluate our technique using extensive real-world traces from a large commercial CDN to show that cluster shutdown can reduce the system-wide energy usage by 67%. Further, much of the energy savings are obtainable without sacrificing either bandwidth costs or end-user performance. In addition, 79% of the optimal savings are attainable even if each cluster is limited to at most one shutdown per day, reducing the required operational overhead. Finally, we argue that cluster shutdown has intrinsic architectural advantages over the well-studied server shutdown techniques in the CDN context, and show that it saves more energy than server shutdown in a wide range of operating regimes.

Keywords: Content Delivery Networks, Internet-Scale Distributed Systems, Power Management, Load Balancing, Energy Efficiency

1. Introduction

Large Internet-scale distributed systems deploy hundreds of thousands of servers in thousands of data centers around the world. Such systems currently provide the core distributed infrastructure for many popular Internet applications that drive business, e-commerce, entertainment, news, and social networking. The energy cost of operating an Internet-scale system is already a significant fraction of the total cost of ownership (TCO) [1]. The environmental implications are equally important. A large distributed platform with 100,000 servers will expend roughly 190,000 MWH per year, enough energy to sustain more than 10,000 households. In 2005, the total data center power consumption was already 1% of the total US power consumption while causing as much emissions as a mid-sized nation such as Argentina. Further, with the deployment of new services and the rapid growth of the Internet, the energy consumption of data centers is expected to grow at a rapid pace of more than 15% per year in the foreseeable future [2]. These factors necessitate rearchitecting Internet-scale systems to include energy optimization as a first-order principle.

An important Internet-scale distributed system to have emerged in the past decade is the *content delivery network* (CDN, for short) that delivers web content, web and IP-based applications, downloads, and streaming media to end-users (i.e., *clients*) around the world. A large CDN, such as that of a commercial provider like Akamai, consists of hundreds of thousands of servers located in over a thousand data centers around the world and account for a significant fraction of the world's enterprise-quality web and streaming media traffic today [3]. The servers of a CDN are deployed in *clusters* where each cluster consists of servers in a particular data center in a specific geographic location. The clusters are typically widely deployed on the "edges" of the Internet in most major geographies and ISPs around the world so as to be proximal to clients. Clusters can vary in size from tens of servers in a small Tier-3 ISP to thousands of servers in a large Tier-1 ISP.

The primary goal of a CDN is to serve content such as web pages, videos, and applications with high availability and performance to end users. The key component that ensures availability and performance is the CDN's *load balancing* system that assigns each incoming request to a server that can serve that request. To this end, a CDN's load balancing system routes each user's request to a server that is *live* and *not overloaded*. Further, to enhance performance, a CDN ensures that each user request is routed to a server that is *proximal* to that user. The proximity (in a network sense) ensures that the network path between the user's device and the CDN's server has low latency and loss. The process of routing user requests to servers is a two stage process. A *global load balancer* (called GLB) assigns the user to a cluster of servers based on the availability of server resources in the cluster, performance, and bandwidth costs. A *local load balancer* (called LLB) assigns the user to a specific server that is capable of serving the requested content within the chosen cluster. The choice of server is dictated by

server liveness, content footprint, and current server loads with respect to their capacities. A comprehensive discussion of the rationale and system architecture of CDNs is available in [3].

1.1. Cluster shutdown: a technique for energy reduction

A number of approaches are relevant to reducing the energy consumption of CDNs. In the past two decades, there has been significant work in improving the energy efficiency of servers and data centers. Such improvements yield energy savings in any deployed distributed system, including CDNs. For instance, the switch to multi-core architectures, the increasing use of SSDs, static power management (SPM) to decrease energy use when the servers are idle, use of low-power servers [4], and power scaling techniques such as Dynamic Voltage and Frequency Scaling (DVFS) [5, 6] all help reduce CDN energy consumption. Similarly, the use of temperature controlled fans and advances in air flow management have led to increases in cooling efficiency[7, 8].

In addition to the above generic methods, there has been recent work on CDN-specific techniques that incorporate the ability to turn off individual servers during periods of low load to reduce the energy consumption [9]. Such a *server shutdown* technique is implemented within the local load balancer (LLB) of the CDN. The work in [9] shows that the availability, performance, and operational costs of the CDN remain unaffected when turning off servers to save energy.

In this paper, we propose and evaluate a novel CDN-specific technique called cluster shutdown where an entire cluster of servers in a CDN data center can be turned off. Cluster shutdown is easily integrated into the global load balancer (GLB) that will now have the ability to move all load away from a cluster and shut it down. However, since the granularity of energy management is to turn off entire clusters or leave them entirely on, the technique does not have the ability to turn off individual servers (e.g., a fraction of a cluster). In contrast, the server shutdown technique studied in [9] has the ability to shutdown individual servers within the cluster depending on the load, but has has no ability to control how much load enters a cluster. Therefore, in this sense, the two techniques are complementary and may be implemented together. While cluster shutdown has not been studied before in the CDN context, it has certain natural advantages that make it worthy of consideration for CDN energy reduction.

(1) Redundant deployments. Large CDNs such as Akamai can have over a thousand clusters deployed in data centers around the world [3] with more than a dozen redundant deployments in any given geographical area. Thus, when some clusters near a user are shutdown during off-peak hours, other nearby active clusters can continue to provide CDN service to users and ensure good availability and performance. In fact, one of the contributions of this work is determining the impact of cluster shutdowns on user performance.

(2) Cluster shutdown is consistent with the original CDN architectural design. Each cluster in a CDN is often architected to be a self-sufficient unit with enough processing and disk storage to serve the content and application domains that are assigned to it [3]. In particular, there is limited data dependency and resource sharing across clusters. Thus, cluster shutdown can be implemented with little or no changes to the CDN's original architecture. In contrast, servers within a cluster are closely linked in a fine-grained fashion and they cooperatively cache and serve the incoming requests. For instance, servers *within* the same cluster cooperatively store application state and content for user requests served by that cluster. Thus, shutting down individual servers for energy savings requires greater migration of state and content between servers in a cluster at levels not customary in a CDN today. Cluster shutdown, in contrast, does not require state migration and cached content is already replicated across clusters for fault-tolerance purposes, which ensures that availability is not impacted by shutting down a cluster. In this sense, cluster shutdown is a better architectural design choice for energy management than server shutdown.

(3) Cluster shutdown has the potential to save on cooling power in addition to IT power. A key advantage of cluster shutdown is that the *all* of the energy consumed by a cluster, which includes energy consumed by the servers, the network equipment, and the cooling within that cluster, can be saved when a cluster is turned off. In contrast, a server shutdown technique will typically turn off a fraction of the servers within the cluster and will require the networking and cooling equipment to stay on. The cooling equipment is not energy proportional—thus turning off a fraction of the servers only saves energy consumed by those servers and does not yield a proportionate reduction in cooling costs.

For cluster shutdown to be effective, a CDN would need to have control over *all* of its energy consumption, i.e., both IT (such as servers) and cooling equipment. Such a scenario is reasonable given the trend for CDN's to opt for self-contained, modular [10], or containerized [11] deployments. With such deployments a CDN can manage the power consumption of its own cluster, independent of other tenants in the data center – an advantage for a CDN that wants manage its power consumption closely. The savings that can be obtained from reducing cooling costs can have a significant impact on the total energy expenditure of a cluster. The key reason is that the energy consumed by cooling equipment is a significant fraction of the energy expended by the IT equipment[1] such as servers. The ratio of total energy to IT energy is a standard metric called PUE (Power Usage Effectiveness) that has a typical value[2] of about 2 implying cooling energy is roughly equal to IT energy in typical data center deployments. But in more recent energy-efficient designs, PUE is smaller but cooling energy is still a significant fraction of the IT energy. Further, cooling energy consumption is *not* power-proportional since cooling still takes a significant amount of energy even when the servers have low utilization and are not producing much heat, resulting in disproportional energy savings when cooling is shutdown entirely (cf. Figure 1a).

[1] IT energy expenditure is primarily the energy consumed by the servers, since the networking equipment consume significantly less. Likewise, cooling energy expenditure is dominated by the energy consumed by the chillers[12].

[2] In a survey by the Uptime Institute [13] in July 2012 , data centers reported an average PUE between 1.8 to 1.89. Other estimates place PUEs even higher.

Despite these advantages, a cluster shutdown technique is not without disadvantages when compared to server shutdown [9]. Shutting down a cluster and moving all its users to other clusters might degrade performance for users if they have to go "farther away" in the network sense for their content. Further, moving traffic across clusters has the potential of increasing the bandwidth cost, even if it reduces energy. A primary focus of our work then is to evaluate the energy reduction provided by cluster shutdown and how it trades off against potential degradation in performance and increases in bandwidth costs.

1.2. Our Contributions

We propose algorithms for incorporating cluster shutdown in the GLB of a CDN and quantify the energy savings achievable by this technique. Our evaluation uses extensive real-world traces collected from 22 geographically distributed clusters over 25 days from one of the world's largest CDNs. We show how energy savings are impacted by the energy characteristics of servers, cooling equipment, and data centers. Further, we quantify the tradeoffs between three goals of CDN architecture: saving energy, reducing bandwidth costs, and enhancing end-user performance. Finally, we compare the relative efficacy of cluster shutdown with the well-studied and complementary approach of server shutdown. Our specific key contributions are as follows.

- We propose a GLB algorithm that minimizes energy by routing traffic away from certain clusters and switching them off. On production CDN workloads with typical assumptions for server and cooling efficiencies, our algorithm achieved a significant system-wide reduction in CDN energy consumption of 67%.
- When servers and cooling equipment are energy inefficient, the energy savings from cluster shutdown can be as large as 73%. These savings can decrease to 61% if the servers become perfectly power proportional, and can further become almost zero if the cooling also becomes perfectly efficient.
- The outside air temperature has an impact on cooling efficiency and hence influences the energy savings achievable by cluster shutdown. Energy savings are stable at about 67% for outside temperatures less than $85^{\circ}F$ but tapers off as the temperature rises to 44% at $100^{\circ}F$.
- To obtain the maximum possible energy savings, bandwidth costs of the CDN would have to increase by a factor of 2. However, 73% of the maximum energy savings are obtainable with no change in bandwidth costs at all. Likewise, 93% of the maximum energy savings is obtainable with no significant performance degradation with each user served from clusters within an average distance of 500 km.
- Frequent cluster shutdowns and the operational overheads that it would entail are not necessary to achieve significant energy savings. Our technique is able to extract 79% of the maximum savings even when limiting each cluster to at most one shutdown per day and even when the incoming load is not known in real-time and must be predicted.
- Realistic CDNs are required to operate under multiple constraints. We identify a sweet spot where our technique provides 22% of maximum savings while limiting each cluster to at most one shutdown per day, allowing no increase in bandwidth costs and serving users from clusters within an average distance of 800 km.
- Cluster shutdown does better than server shutdown within a broad operating range of outside air temperatures from $40^{\circ}F$ to $90^{\circ}F$, while server shutdown is better outside of this range. In general, cluster shutdown performs better during lower periods of CDN utilization, while server shutdown has the edge at higher utilization.
- Augmenting cluster shutdown with server shutdown has limited impact under relaxed performance or bandwidth constraints because the CDN is already nearly power proportional under these conditions with just cluster shutdown. However, if either latency or bandwidth costs need to be kept low, server shutdown can provide significant additional gains over a pure cluster shutdown strategy. If low latency is required, server shutdown can provide an additional 46% in energy savings. Likewise, if no increase in bandwidth costs are allowed, the additional energy savings is 34%.

2. Background, Models, and Methodology

2.1. Content Delivery Networks

Our work assumes a global content delivery network (CDN) that comprises a very large number of servers that are grouped into thousands of clusters. Each cluster is deployed in a single data center and its size can vary from tens to many thousands of servers. The incoming requests are forwarded to a particular server in a particular cluster by the CDN's load balancing algorithm. As outlined earlier, load balancing in a CDN is performed in two stages: global load balancing (GLB) that routes a user's request to an "optimum" cluster, and local load balancing (LLB) that assigns the user request to a specific server within the chosen cluster. Load balancing can be implemented using many mechanisms such as IP Anycast, load balancing switches, or most commonly, the DNS lookup mechanism [3]. We do not assume any particular mechanism, but we do assume that those mechanisms allow load to be arbitrarily re-divided and re-distributed among servers, both within a cluster (local) and across clusters (global). This is a good assumption for typical web workloads that form a significant portion of a CDN's traffic.

Our proposed technique of cluster shutdown is implemented in the GLB of a CDN. First, GLB moves away all the traffic from a cluster, typically by setting the cluster capacity to zero. Then, the cluster is shutdown by turning off all the relevant components, inclusive of servers and cooling equipment. Since our focus is on GLB algorithms that incorporate cluster shutdown, unless mentioned otherwise, we assume that the LLB evenly distributes the incoming load assigned by the GLB across servers within that cluster. In contrast, the server shutdown mechanism studied in [9] is incorporated within the LLB system that turns off individual servers within a cluster.

2.2. Workload Model

The workload entering a CDN is generated by users around the world accessing web pages, video content, and Internet-based applications. To model the spatial distribution of the users, we cluster them according to their geographical location. In particular, we define *M client locations* where each location is a compact geographical area, example, Massachusetts, USA. The workload entering the CDN is modeled as a discrete sequence[3] $\lambda_{t,i}$, $1 \leq t \leq T$ and $1 \leq i \leq M$, where $\lambda_{t,i}$ is the average load in the t^{th} time slot from users in client location i. We always express load in the normalized unit of actual load divided by peak server capacity.[4] Further, we assume that each time slot is δ seconds long and is large enough for the decisions made by the global load balancing algorithm to take effect. Specifically, in our experiments, we consider $\delta = 5$ minutes.

2.3. Algorithmic Model for Load Balancing

While a real-life load balancing system is complex [3], we model only those aspects of such a system that are critical to energy usage. For simplicity, our load balancing algorithms redistribute the incoming load rather than explicitly route incoming requests from clients to servers. The major determinant of energy usage is the number of clusters that need to remain active (i.e., turned on) at each time slot to effectively serve the incoming load. Unless we mention otherwise, we assume that local load balancer is not energy aware and does not turn servers on and off on its own accord. But, rather, the LLB simply distributes the load assigned to each cluster evenly among the servers in that cluster. However, the GLB is energy aware and can turn clusters on or off. Therefore a cluster is either active with all servers turned on, or inactive with all servers turned off.

At each time slot, an energy aware GLB takes as input the incoming load λ_i, $1 \leq i \leq M$. The global load balancing algorithm of a CDN routes the incoming load from each client location i to clusters that are active at that time step, i.e., GLB determines the values μ_{ij} that represents the load induced by client location i on a server in the j^{th} cluster, $1 \leq j \leq N$, such that

$$\sum_{1 \leq j \leq N} \mu_{ij} c_j = \lambda_i, \forall i,$$

where c_j is the number of servers in that cluster. Servers are typically not loaded to capacity. But rather a *target load threshold* μ_{max}, $0 < \mu_{max} \leq 1$, is set such that the load balancing algorithm attempts to keep the load on each server of the CDN to no more than μ_{max}. Mathematically,

$$\sum_{1 \leq i \leq M} \mu_{ij} \leq \mu_{max}, \forall j.$$

We assume a typical value of $\mu_{max} = 0.75$ in our work, i.e., the target load for each server is 75% of its capacity.

[3] When the time slot is implicit, we often drop the time subscript from our notation. For instance, we describe the incoming load simply λ_i, $1 \leq i \leq M$.

[4] For simplicity, we assume that the servers in the CDN are homogeneous with identical capacities, though our algorithms and results can be easily extended to the heterogeneous case.

2.4. Power consumption of clusters

We model both the power consumed directly by the servers (IT power) and the power consumed for cooling those servers (cooling power). By convention, we indicate power draw for a single server by using a superscript "server", while variables without that superscript represent the power draw for the entire cluster. Also, note that while we mostly discuss power draw (in Watts), integrating power draws over time provides us the energy consumed (in Joules).

2.4.1. Server power model

First, we model the power consumed by a single server as a function of its load. Based on our own testing of typical off-the-shelf server configurations used by CDNs, we use the standard linear model[1] where the power (in Watts) consumed by a server is

$$P^{\mathbf{IT,server}} = \left[P_{idle}^{\mathbf{IT,\ server}} + (P_{peak}^{\mathbf{IT,\ server}} - P_{idle}^{\mathbf{IT,\ server}})\lambda \right] \quad (1)$$

where the load ($0 \leq \lambda \leq 1$) is the server load, and $P_{peak}^{\mathbf{IT}}$ is power consumed by the server when it is loaded to its capacity (i.e., $\lambda = 1$). $P_{idle}^{\mathbf{IT,server}}$ is the power consumed by an idle server when it has no load (i.e., $\lambda = 0$). We define a quantity $0 \leq \alpha \leq 1$ called the *server power proportionality factor* where

$$\alpha \stackrel{\Delta}{=} 1 - P_{idle}^{\mathbf{IT,\ server}} / P_{peak}^{\mathbf{IT,server}}.$$

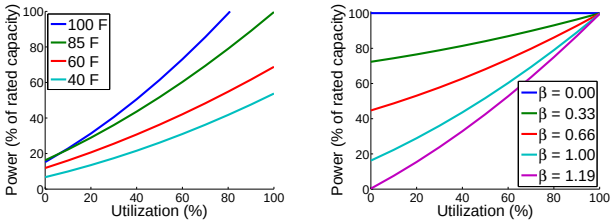
Note that $\alpha = 1$ represents a perfectly power proportional server—the ideal case for an energy-efficient server—while $\alpha = 0$ represents the opposite extreme. In our empirical work, unless mentioned otherwise, we use $P_{peak}^{\mathbf{IT,server}} = 92$ Watts, α =0.31, and $P_{idle}^{\mathbf{IT,server}} = 63$ Watts as typical values based on our measurements of a typical deployed server today. However, we also vary α over a wide range to study the impact of server power proportionality on our conclusions.

2.4.2. Cooling power model

The cooling systems deployed to cool a server cluster consist of a number of components. An air handler transfers heat out of the server room. An air or water based chiller cools down the hot air before it is pumped back into the server room. The coolant, usually a combination of water and glycol[5] is transferred from the chiller to cooling towers that exchange heat with the outside air before returning it back to the chiller. The chiller plant's compressor accounts for the majority of the cooling cost in most data centers [12].

To make our model assumptions realistic, we use a set of benchmark regression curves provided by the California Energy Commission (CEC) [14] to model our cooling power requirements. Assuming efficient heat exchange at the cooling towers, we take the outside air temperature as a proxy for the temperature of the coolant on return. The power consumed by the chiller $P^{\mathbf{COOL}}$ is a *quadratic* function of its utilization u as

[5] For the purposes of modeling a typical cooling system, we assume that the chilled water coolant is at $44°F$.



(a) The chiller power P^{COOL} gets larger and steeper as outside temperature increases.

(b) As cooling efficiency β increases the cooling power required $P^{\text{COOL}}(\beta)$ is smaller.

Figure 1: Cooling power and its dependence on outside air temperature and cooling efficiency.

shown below [14], where $u \triangleq \frac{Q}{Q_{peak}}$, Q is the heat removed by the chiller, and Q_{peak} is maximum heat removal that the chiller is rated for.

$$P^{\text{COOL}} = P^{\text{COOL}}_{peak} \times \left(A + B \cdot u + C \cdot u^2\right) \tag{2}$$

where u is the utilization of the chiller and the constants A, B, and C are dependent on the capacity correction factor (CAP_FT) and the efficiency correction factor (EIR_FT) that vary quadratically with the outside air temperature. Given a value for the outside air temperature the constants A, B, and C can be derived from the regression curves provided in the CEC manual [14]. *It is worth noting that a chiller consumes disproportionately more power at higher utilization than lower ones due to the quadratic nature of the curve.* Also, as shown in Figure 1a, as the outside air temperature gets higher the power required P^{COOL} gets larger and curve becomes more non-linear and steeper.

The chillers deployed in practice vary greatly in terms of their efficiency, ranging from less efficient older systems to highly efficient next-generation ones. To study this wide variation, we propose a family of chiller models that have the same quadratic functional form for the relationship between utilization and power consumed as the CEC chiller described in Equation 2 but different values for the constants. Specifically, we define a factor β that we call the *chiller efficiency factor* and each value of β provides a different curve for the chiller power consumption $P^{\text{COOL}}(\beta)$ as described in the equation below.

$$P^{\text{COOL}}(\beta) = P^{\text{COOL}}_{peak} \times \left(A_\beta + B_\beta \cdot u + C_\beta \cdot u^2\right), \tag{3}$$

where $A_\beta = \max\{(\beta A + 1 - \beta), 0\}$, $B_\beta = \beta B$, and $C_\beta = \beta C$.

We study five chillers by setting β to five different values as shown in Figure 1b. The first three curves ($0 \leq \beta < 1$) represents chillers that are less efficient than CEC's chiller. As can be seen from Equation 3, the fourth curve with $\beta = 1$ models the CEC chiller exactly. And, the fifth curve with $\beta > 1$ models a next-generation chiller that is more efficient than the CEC chiller and has power consumption of zero when idle.

2.4.3. Total power consumed by a cluster

The total power consumed by a cluster is defined to the power needed to run the servers and associated equipment and the power needed to cool the servers. We define the IT power P^{IT} of a cluster to be the aggregate power consumed by the c servers of the cluster. In addition to the servers themselves, a cluster includes other IT equipment such as network switches and power distribution units. Typically the power consumed by networking and power distribution equipment is a small fraction of that consumed by the servers of the cluster (studies have shown this portion to be around 5-10% [12]). Our power model currently ignores the contribution of this other IT equipment to the total IT power, but it is straightforward to extend our models and algorithms to incorporate its contribution through a small multiplicative constant.

Thus, the IT power P^{IT} consumed by a server cluster consisting of c servers, each running at utilization λ, is

$$P^{\text{IT}} = c \times P^{\text{IT,server}} \tag{4}$$

where $P^{\text{IT,server}}$ can be computed using Equation 1. And, the peak IT power of a cluster $P^{\text{IT}}_{peak} = c \times P^{\text{IT,server}}_{peak}$. Given the PUE of the data center in which the cluster is deployed, we determine the peak cooling power $P^{\text{COOL}}_{peak} = (\text{PUE} - 1) \times P^{\text{IT}}_{peak}$. Since the chiller removes the heat produced by the servers, the utilization of the chiller $u = \frac{P^{\text{IT}}}{P^{\text{IT}}_{peak}}$. Now, given the value of β that determines the cooling efficiency, we can compute the total power consumed by the chiller $P^{\text{COOL}}(\beta)$ using Equation 3. The total power P consumed by the cluster is simply the sum of its IT and cooling power:

$$P = P^{\text{IT}} + P^{\text{COOL}}(\beta) \tag{5}$$

Note that the quadratic and non-energy proportional nature of the chiller-based cooling model has interesting implications on cluster and server shutdown techniques. When a server shutdown technique switches off a fraction of the servers within a cluster, the non-energy proportional nature of the curve works "against" it and does not yield a proportional reduction in cooling energy usage, while a full cluster shutdown reduces the cooling costs to zero for that cluster. In contrast, cluster shutdown "packs" the load from a region onto a smaller number of clusters (and turns off the remaining clusters), but the quadratic nature of the curve yields *more than linear* increase in cooling costs for the clusters that stay on; the higher the cluster utilization due to such packing, the greater the increase in cooling cost due to the quadratic nature of the curve. A similar effect comes into play due to the outside air temperature, where increasing cluster utilization in hotter outside temperature causes a disproportionately larger increase in cooling costs due to the quadratic curve.

3. GLB Algorithms with Cluster Shutdown

We now describe our algorithm for global load balancing that routes traffic from client locations to clusters while turning clusters on or off with the goal of minimizing the total energy consumed by the CDN. At any given time, the algorithm takes as input the load λ_i from each individual client location i. Here we make the simplifying assumption that the GLB knows precisely the load that it needs to route at each time step and that

it can instantaneously turn clusters on or off to minimize energy usage. This is clearly not strictly true in practice where both sensing the load and shutting down clusters incur a small delay. However, our algorithm provides a baseline on what is achievable with the cluster shutdown technique, leaving a more complex model that incorporates delays for future work. The output of our algorithm is two-fold. First, it computes a binary variable u_j that indicates whether the j^{th} cluster should be turned on ($u_j = 1$) or turned off ($u_j = 0$) in that time step. Next, it computes a quantity μ_{ij} that represents the load from client i that must be routed to cluster j at the given time step.

Computing the assignment of load to clusters can be stated as a convex optimization problem as follows. The IT power required by cluster j is

$$P_j^{\mathbf{IT}} = c_j \left[P_{idle}^{\mathbf{IT,server}} \times u_j + \sum_{1 \le i \le M} (P_{peak}^{\mathbf{IT,server}} - P_{idle}^{\mathbf{IT,server}}) \mu_{ij} \right],$$

where the value of u_j is used to determine if the cluster is turned on and idle power should be incurred. The chiller utilization of cluster j can be computed as $\frac{P_j^{\mathbf{IT}}}{P_{peak}^{\mathbf{IT}}}$. The corresponding cooling energy for cluster j denoted by $P_j^{\mathbf{COOL}}$ can be computed using Equation (3), given the chiller efficiency β. Our objective function is simply the total power drawn by the CDN summed across all its N clusters and is stated below.

$$\min \sum_{1 \le j \le N} \left(P_j^{\mathbf{IT}} + P_j^{\mathbf{COOL}} \right) \tag{6a}$$

$$s.t. \sum_{1 \le j \le N} \mu_{ij} c_j = \lambda_i, \qquad \forall i \tag{6b}$$

$$\sum_{1 \le i \le M} \mu_{ij} \le u_j \mu_{max}, \quad \forall j \tag{6c}$$

The quantities that are varied in the minimization are the output variables $\mu_{i,j}$ and u_j. Equation 6b ensures that the all of the incoming load at the given time step is assigned to some cluster. Further, Equation 6c ensures that no server is loaded by more than the threshold μ_{max}. We pick a typical value of $\mu_{max} = 0.75$ that implies that no server is loaded to more than 75% of its capacity.

Besides the above constraints that always apply, we also study tradeoffs between energy savings, performance and bandwidth costs by adding one or both of the constraints below.

$$\frac{\sum_{1 \le i \le M} \sum_{1 \le j \le N} \mu_{ij} c_j d_{ij}}{\sum_{1 \le i \le M} \lambda_i} \le D, \tag{7a}$$

$$\sum_{1 \le i \le M} B_i \frac{\mu_{ij} c_j}{\lambda_i} \le BW_{max}(j), \quad \forall j \tag{7b}$$

Equation (7a) states that the average distance between the users and the cluster to which they are assigned (weighted by load) is no more than some specified value D, where d_{ij} is the geographical distance between client location i and cluster j. For smaller values of D, this equation constrains the global load balancer to assign users to server clusters that are proximal to them, so as to ensure good performance. By making D larger, we are loosening the performance requirement by allowing the users to be assigned to clusters that are farther away. We are particularly interested in how the performance requirement D impacts energy savings. Note that as was assumed in earlier work [15], we use geographical proximity as a rough proxy for "network proximity" that governs user performance. Our formulation could equally well accommodate network latency instead of geographical distance without significant changes, though our empirical CDN traces do not provide the required network information for such an evaluation.

A CDN pays for the bandwidth that their deployed servers utilize. Typically, CDNs use 95/5 contracts for paying for their bandwidth use which works as follows [16]. For each cluster j, the traffic from the CDN's servers in the cluster is averaged over 5 minute intervals. Then the 95^{th} percentile of those 5-minute averages over the billing interval is computed. The cost of bandwidth for that cluster is proportional to the computed 95^{th} percentile. Since 95^{th} percentile cannot be modeled and constrained within a convex programming framework, we use the maximum value instead as a proxy. Equation (7b) above is used to constrain the maximum bandwidth sent out of cluster j to be no more than $BW_{max}(j)$, hence also constraining the bandwidth cost that is incurred by the CDN in cluster j. Choosing higher values for $BW_{max}(j)$ is tantamount to increasing the allowable bandwidth cost at cluster j. We use the bandwidth constraint to study the impact of varying the bandwidth costs on energy savings.

Converting the convex program to a mixed integer program (MIP). Note that as currently stated the objective of the optimization function in Equation 6 contains the term $P_j^{\mathbf{COOL}}$ that is quadratic in variables μ_{ij}. However, since MIPs are more tractable than convex programs, we used a linear piecewise approximation of $P_j^{\mathbf{COOL}}$ to rewrite the optimization with only linear constraints. The domain for the function $P_j^{\mathbf{COOL}}(u)$ was split into equal sized segments. For each such segment $[x_i, x_{i+1}]$ we sampled the value of the function at its endpoints $[y_i, y_{i+1}]$. Computing the slope m_i and intercept k_i, the linear approximation between the points (x_i, y_i) and (x_{i+1}, y_{i+1}) takes the form $P_{j,(x_i,x_{i+1})}^{\mathbf{COOL}}(u) = P_{j,peak}^{\mathbf{COOL}} \times (m_i \cdot u + k_i)$. For each such segment we added a constraint

$$P_j^{\mathbf{COOL}} \ge P_{j,(x_i,x_{i+1})}^{\mathbf{COOL}}$$

with cluster j running at a chiller utilization of $u = \frac{P_j^{\mathbf{IT}}}{P_{peak}^{\mathbf{IT}}}$. $P_j^{\mathbf{COOL}}$ is present in the objective and lower bounded by the piecewise linear approximation. The absence of any other constraint on the variable ensures that it equals its lower bound in the optimal solution. Our implementation used 5 linear segments for an approximation error of 0.25% at $85°F$.

4. Combining Cluster and Server Shutdown

Server shutdown is a complementary technique to cluster shutdown and turns off individual idle servers within each cluster to save energy [6, 9]. We now devise a combined approach of using server shutdown in conjunction with our cluster shutdown algorithm to potentially provide even more energy savings. Our combined approach first explores the possibility of

shutting down entire clusters, thereby saving both the IT and cooling energy consumed by those clusters. Note that a cluster shutdown algorithm must maintain a distributed set of clusters in an active state at all times for reasons of user-perceived performance. For instance, if all clusters in a geographical region are shut down, GLB will be forced to assign users from that region to distant clusters resulting in larger latencies and degraded performance. Server shutdown can provide additional energy savings within clusters that are kept active by the cluster shutdown algorithm. In particular, not all of the servers in an active cluster may be required to serve its assigned load and a subset of these servers can be turned off to save more energy.

To capture the additional benefit of server shutdown, we enhance the cluster shutdown algorithm of Section 3 by incorporating server shutdown algorithms within the LLB of individual server clusters. We propose a hierarchical strategy that consists of the following two steps.

1. GLB decides which clusters should remain active and which need to be turned off using the algorithm described in Section 3. GLB then reroutes global traffic away from clusters that can be turned off and reassigns that traffic to clusters that remain active.
2. The server shutdown algorithm is run independently and in parallel by the LLB in each active cluster at each time step. For each active cluster, the LLB of that cluster consolidates the load assigned to that cluster into the fewest number of servers possible and turns off the remaining servers. Specifically, for a cluster of c servers, a target load threshold μ_{max} and load λ, LLB computes the optimal number of live servers $c_t = \left\lceil \frac{\lambda}{\mu_{max}} \right\rceil$ that is required to serve the load. The algorithm keeps $c - c_t$ servers inactive while keeping c_t servers active to serve the load λ.

In step (2) of our above algorithm, we make the simplifying assumption that servers can be shutdown in one time step, providing a baseline for the savings possible. A more complex server shutdown algorithm that takes into account the delay for transitioning servers between active and inactive states is provided in [9].

5. Evaluation

To evaluate the benefits of integrating cluster shutdown in a CDN's global load balancer we used extensive traces from Akamai, perhaps the largest commercial CDN, and ran the algorithms presented in Section 3. In our experiments, unless otherwise indicated, we model chillers with $\beta = 1$, i.e., the same as CEC's chiller model, and we assume that the outside air temperature is $85°F$. Later, we vary these parameters and show how energy savings vary with different parameter values.

5.1. Empirical Data from the Akamai Network

We used extensive load traces collected over 25 days from a large set of Akamai clusters deployed in data centers in the US. The 22 clusters captured in our traces are distributed widely within the US and had 15439 servers in total, i.e., a representative sampling of Akamai's US deployments. Our load traces

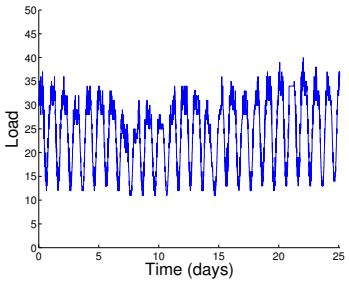
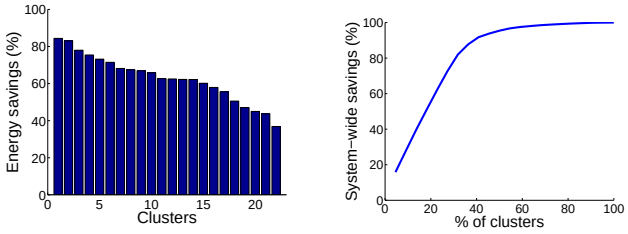


Figure 2: Average load per server measured every 5 minutes across 22 Akamai clusters in the US over 25 days.

account for a peak traffic of 800K requests/second and an aggregate of 950 million requests delivered to clients. The traces consist of a snapshot of total load served by each cluster collected every 5-minute interval from Dec 19th 2008 to January 12th 2009, a time period that includes the busy holiday shopping season for e-commerce traffic (Figure 2). In the figure, one may note load variations due to day, night, weekday, weekend, and holidays (such as low load on day no. 8, which was Christmas) Since the clusters are restricted to the US, we also restricted the trace to clients from North America. The trace consists of samples taken every 5 minutes indicating the current load on each cluster, along with a breakup of traffic from each client location. Specifically, for every 5 minutes, we measured the load induced by client location i on cluster j and the corresponding bytes served by cluster j to users in client location i, for all relevant pairs of i and j. In addition, we also measured the number of servers present and total capacity of each cluster. In the course of our optimization, we assume that the load from a client can be shifted to any cluster as long as the capacity constraints are met and no server is overloaded. Our traces also capture the geographic location (city, state, and country) of both the client location and cluster, which lets us estimate the geographical distance between the users at a particular client and location the cluster from which they are served. The geographical distance computed in this fashion is used as a proxy for performance. The byte information captured in our traces is used to compute the bandwidth usage of the CDN in each cluster that in turn determine the bandwidth costs incurred by the CDN that we study in our work.



(a) Individual clusters save between 37% to 84%. The system-wide energy savings is 67%.

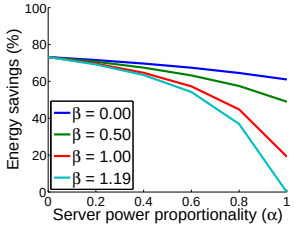
(b) Applying cluster shutdown to the top 45% clusters is sufficient to obtain 94% of the system-wide energy savings.

Figure 3: CDN energy savings obtainable by cluster shutdown.

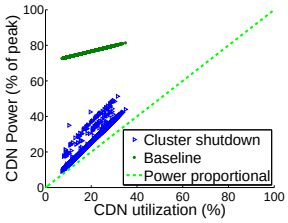
5.2. Overall energy savings

We emulated the GLB-based cluster shutdown algorithm in Section 3 on the CDN traces described above. The algorithm minimizes the energy consumption of the CDN in each time step by orchestrating which clusters should be on and which clusters should be turned off. Then the total energy consumed by the CDN is computed by adding the energy consumed at each time step across the entire trace. As a basis for comparison, we used as a baseline the energy consumed by the user-to-cluster assignment in the trace with no cluster shutdown, i.e., all clusters are assumed to be on throughout the trace which is consistent with how CDNs operate today.

The system-wide energy savings that is possible with cluster shutdown incorporated into the CDN's GLB is 67% in comparison with the baseline where all clusters are always turned on. In performing this analysis, we make typical assumptions about the energy efficiency of the data centers (PUE = 2), servers ($\alpha = 0.31$) and chillers ($\beta = 1$). We also do not constrain performance and bandwidth costs. Therefore, these are the best case savings possible. However, we vary each of these assumptions in subsequent sections to examine how these savings change under different scenarios. To further breakdown the savings, in Figure 3a we show savings obtained by individual server clusters. Savings vary between 37% to 84% with the median cluster saving 63%. Further, most of the savings can be obtained by performing cluster shutdown in a few key clusters. As shown in Figure 3b, applying cluster shutdown to top 45% of the clusters is sufficient to obtain 94% of the optimal energy savings.



(a) Energy savings decrease as servers and cooling equipment become more energy efficient.



(b) Cluster shutdown makes the CDN power proportional by aligning power values close to the ideal 45-degree line.

Figure 4: Energy savings and power proportionality

5.3. Impact of server and cooling efficiency

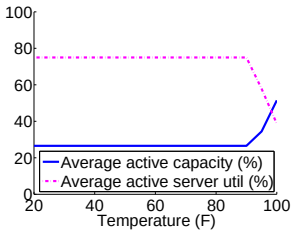
CDNs operate with a wide range of server hardware and are deployed in a wide range of data center facilities. Further, both server and cooling efficiencies are constantly being improved over time. To capture these effects, we varied the power proportionality factor of the servers (α) as well as the cooling efficiency of the chillers (β) to study how energy savings vary with these parameters (cf., Figure 4a). When both the servers and cooling are energy-inefficient ($\alpha = \beta = 0$), the cluster shutdown technique provides the most energy savings of 73%.

As servers become more energy-efficient the idle power usage gets lower, and thus lowers cooling energy. This results in energy savings from cluster shutdown dropping to 61% when servers are perfectly power proportional ($\alpha = 1$). In fact for any chiller efficiency β, energy savings decrease as servers become more efficient.

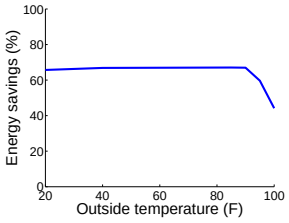
Likewise, for any given server efficiency α, increasing cooling efficiency β reduces the energy savings. For perfectly power proportional servers ($\alpha = 1$) energy savings fall as β increases, dropping from 61% when $\beta = 0$ to 19% for $\beta = 1$. In the ideal world with highly-efficient servers and cooling, e.g., $\alpha = 1$ and $\beta > 1$, the energy savings from cluster shutdown approaches zero, i.e., if the "hardware" is itself highly-efficient there is no need for an explicit shutdown mechanism to reduce energy.

5.4. CDN Power Proportionality

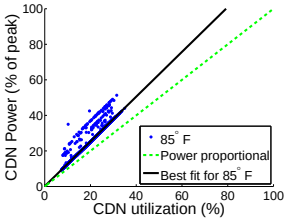
To visualize how server shutdown makes a CDN more power proportional, it is instructive to view the instantaneous power consumption of the entire CDN as a function of its overall utilization. Specifically, in Figure 4b, we plot the CDN's total power consumption (as a percentage of its peak) and its overall utilization at each time step as a single point of a scatter plot. Note that these plots are the exact analogue of server proportionality described in Equation 1 that relates power to utilization, but computed for the CDN as a whole. A perfectly power proportional system would have all its points aligned along the 45-degree line shown in the figure. The scatter plot of the total CDN power without cluster shutdown deviates from the ideal 45-degree line significantly as the CDN consumes a lot of power even during periods of low utilization during the non-peak hours. However, cluster shutdown makes the scatter plot of the total CDN power much more closely aligned to the ideal 45-degree line, i.e., cluster shutdown makes the CDN significantly more power proportional.



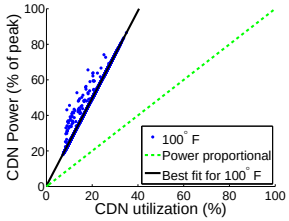
(a) Avg. active server utilization falls as temperature rises



(b) Energy savings drop from 67% at 85°F to 44% at 100°F



(c) At 85°F, the CDN with cluster shutdown is roughly power proportional.



(d) At 100°F, cluster shutdown is less effective.

Figure 5: Cluster shutdown is more effective in saving energy at lower temperatures than higher ones.

5.5. Impact of Outside Air Temperature

The cooling equipment transfers heat from inside the server room to the external atmosphere. Physical laws suggest that the heat transfer rate through convection is larger when the temperature differential between the inside and outside air temperatures are greater. Thus, it takes less energy to cool when the outside temperature is cooler (say, in the winter) than when the outside temperature is hotter (say, in the summer). Further, as we saw in Figure 1a, the required power for cooling rises more sharply in a quadratic fashion with increasing utilization when the outside air temperature is hotter.

The interplay of outside air temperature with cooling power impacts what energy savings are achievable by GLB via cluster shutdown. Specifically, as outside air temperature increases, the cluster (and server) utilization have to be kept low since there is a greater cooling power penalty associated with higher utilization. Thus, as shown in Figure 5a, at low temperatures the algorithm runs all active servers at the maximum allowed utilization of $\mu_{max} = 75\%$. At high temperatures cooling costs rise rapidly with utilization, and the optimal solution at $100°F$ corresponds to active servers running at 39% utilization. Note that to continue to serve the same incoming load, a lower cluster (or, server) utilization means more clusters (and, servers) need to remain active. Thus, the fraction of total CDN capacity that is kept active, rises from 27% at low temperatures, to 51% of total capacity at $100°F$. The increase in active capacity with rising temperatures combined with lower utilization of active servers has a negative impact on savings. Figure 5b shows that energy savings drop from 67% at $85°F$ to 44% at $100°F$. The energy savings achieved by cluster shutdown at different outside air temperatures can also be viewed as a scatter plot of the total CDN power versus its utilization. The scatter plots in Figures 5c and 5d correspond to $85°F$ and $100°F$ respectively. At $85°F$ the best linear fit to the power-utilization curve has a slope of 1.26, closer to the ideal 45-degree line with a slope of 1, i.e., the CDN with cluster shutdown is roughly power proportional. At $100°F$ the slope almost doubles to 2.46.

5.6. Tradeoff between Energy and Performance

CDNs host a wide range of applications. Some applications such as dynamic web sites are highly sensitive to network latency, with even small increases in latency causing significant degradation in the performance experienced by the user. Other applications such as software downloads are weakly sensitive to latency and can even be performed in the background. As in [15], we use geographical distance as a rough proxy for the network latency between a user and the cluster assigned to that user by GLB. To study the tradeoff between performance requirement and energy savings we add Equation (7a) as a constraint where different latency requirements can be modeled by varying the distance bound D. Specifically, larger values of D allow a larger load-weighted average distance between the users and their assigned clusters. Allowing larger user-cluster distances (and latencies) has the effect of degrading performance, but allows for potentially more cluster-shutdown opportunities for GLB and greater power savings. Figure 6 illustrates this tradeoff where setting $D = 300$ km provides 46% of optimal savings. Note that this distance bound is roughly the distance between Boston and New York with network latencies often in the 10-15 ms range that is adequate for even applications with higher latency sensitive. When $D = 500$, one can achieve 93% of the energy savings. This distance bound is roughly the distance between Boston and Philadelphia where typical latencies are in the 20 ms range, suitable for most moderately latency-sensitive applications. Finally, when $D = 795$ km, a suitable limit for weakly latency-sensitive applications such as background downloads, we achieve 99.9% of optimal savings.

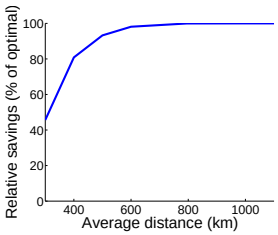


Figure 6: Relaxing performance results in greater energy savings. 46%, 93% and 99.9% of the optimal energy savings are obtained at D values of 300 km, 500 km and 795 km respectively

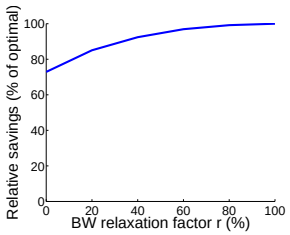
5.7. Tradeoff between Energy and Bandwidth Costs

The operating expenditure (OPEX) of a CDN includes two major components: the energy costs for powering the servers and the bandwidth cost for the traffic from the server clusters to the users. Reducing energy usage by packing traffic into fewer server clusters could cause increased bandwidth usage in those clusters, which in turn could drive up the bandwidth cost at those clusters. The primary question is whether energy savings can be achieved without significant increase in the bandwidth cost. Note that if energy savings are only obtainable by significantly increasing the bandwidth cost, that would serve as a disincentive for a CDN to implement cluster shutdown.

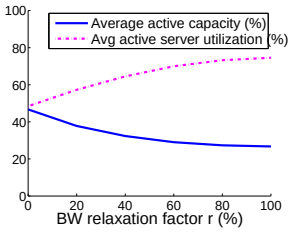
As noted in Section 3, the bandwidth cost incurred by the CDN at each cluster can be approximated by the maximum over all 5-minute time slots in the billing period[6] of the average traffic (in Mbps) transmitted in that time slot. We constrain (through Equation (7b)) the maximum bandwidth for each cluster j to be at most $(1 + r)BW_{max}(j)$, where $BW_{max}(j)$ is the maximum bandwidth value observed in the trace and r is the BW relaxation factor that determines how much extra bandwidth costs we are willing to allow. Figure 7a shows energy savings relative to optimal as the bandwidth constraints are relaxed by varying r. With no increase in bandwidth cost ($r = 0$), cluster shutdown can still achieve 73% of optimal savings. 47% of the total CDN server capacity remains turned on, with active servers running at an average utilization of 48%. Relaxing bandwidth constraints allows active server utilization to rise to $\mu_{max} = 75\%$ at $r = 100\%$. This allows the CDN to run with 27%

[6] In our simulations, we assume that the billing period is length of the trace which is 25 days, though in reality a billing period is typically one month.

of its server capacity turned on and achieve optimal energy savings. Overall, our results indicate that cluster shutdown can still achieve significant energy savings with little or no increase in bandwidth costs.

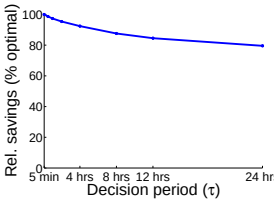


(a) We get 73% of optimal energy savings with no increase in bandwidth cost

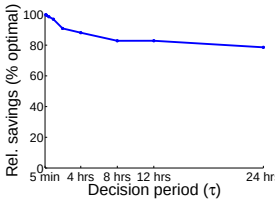


(b) Average active server utilization increases from 49% to μ_{max} = 75% as BW cost doubles (r=100%)

Figure 7: Energy savings versus Bandwidth cost



(a) Switching clusters once a day still achieves 80% of optimal savings



(b) With load prediction we achieve 79% of optimal savings switching clusters once a day

Figure 8: Impact of decision period and traffic prediction

5.8. Impact of Limiting the Cluster Transitions

Frequently switching server clusters on and off can impact the overall lifetime and reliability of the equipment. Further, the mechanical nature of cooling equipment limits the rate at which it can be switched on and off. Chillers, for example, require a warm up at partial load before they can be incrementally ramped up to full capacity. Thus it is neither desirable nor feasible to frequently turn entire clusters on and off, and we study the amount of energy savings that can be extracted when limiting the frequency of cluster shutdowns.

Suppose that cluster transitions are allowed to occur only once every τ time slots, where τ is defined as the *decision period* and is required to be an integral multiple of δ. In our experiments we vary τ from 5 minutes to 1 day. In Figure 8a the left-most point in the graph corresponds to $\tau = 5$ minutes which is the smallest time granularity at which the trace data is collected. It is nearly infeasible to turn clusters on or off every 5 minutes. However, the $\tau = 5$ minutes measurement provides the theoretical optimal of how much energy savings is possible in the best case that can serve as a benchmark for comparing other values of τ. Increasing τ could decrease energy savings as GLB has a lesser ability to turn clusters on or off in response to load variations. However, as we see in Figure 8a, even with $\tau = 1$ day where clusters are transitions just once a day, we achieve 80% of the optimal savings possible. Thus, we establish that frequent cluster transitions are not necessary for obtaining most of the benefits of cluster shutdown.

5.9. Impact of inaccurate real-time load information

Thus far, we have assumed that the load for the current decision period τ is accurately available and can be used for decision making for that period. This is a reasonable assumption for smaller decision periods (say $\tau \leq 30$ minutes) but not so much when the decisions are more infrequent and decision periods are longer. Therefore we consider the situation where our algorithm does not know the current load but would have to predict it for the purpose of deciding which clusters are transitioned. When cluster transitions are made based on a prediction of load over any extent of time there always exists the chance of insufficient active capacity and users being denied service. We allow active CDN clusters to run to 100% utilization before they drop incoming workload. We define *availability* as the ratio of workload served to total workload. Under these assumptions, we define a simple algorithm that predicts the load and computes the optimal cluster allocation under this prediction. The predicted load equals the load at the previous decision period, for small decision periods ($\tau \leq 1$ hour), or the load at the same decision period from the previous day, for larger periods ($\tau > 1$ hour). Using this simple prediction algorithm, Figure 8b shows energy savings for decision period 5 minutes $\leq \tau \leq 1$ day. Energy savings dropped from 100% to 79% of optimal over this range. In each case, the algorithm provided at least "three nines" of availability (i.e. 99.9%).

5.10. Finding a sweet-spot

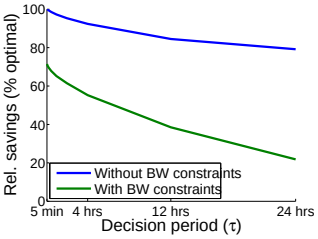


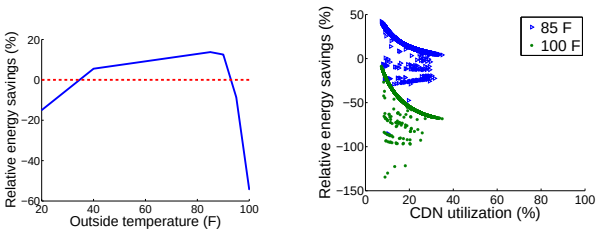
Figure 9: We can achieve 22% of the optimal savings even with switching each cluster no more than once a day, allowing no increase in bandwidth costs, and limiting the average distance from the user to the cluster to be no more than 800 km.

So far we looked at the impact of individual parameters on the energy savings obtained through cluster shutdown. In a realistic situation, we would expect CDNs to operate under multiple constraints. In this section we look at the combined impact of cluster transitions, performance and bandwidth constraints on energy savings. Figure 9 shows energy savings as a function of the decision period when the average user-cluster distance is upper bounded at $D = 800$ km. With no increase in bandwidth costs (corresponding to $r = 0$), for a decision period (τ)

of 5 minutes, and a performance constraint of 800 km we obtain 71% of optimal savings. This compares favorably with the 73% savings without the performance constraints (Section 5.7). Savings fall to 22% of optimal as the decision period (τ) increases to 1 day.

5.11. Cluster vs Server shutdown

We look at the relative energy savings of two complementary techniques: GLB that incorporates cluster shutdown and an LLB that incorporates server shutdown. We assume that, given a cluster with c servers getting incoming load λ, LLB always keeps the exact number of servers $\lceil \lambda/\mu_{max} \rceil$ required to serve the incoming load for that cluster and at every time step. This is of course an optimistic assumption but it helps understand the best possible savings achievable using LLB. However, unlike GLB, LLB is unable to move traffic across clusters to shutdown entire clusters. Figure 10 plots the difference between the energy savings of implementing cluster shutdown in GLB and the corresponding savings from implementing server shutdown in LLB. In Figure 10a, we see that at low outside air temperatures when cooling is relatively inexpensive (cf., Fig 1a), LLB with server shutdown performs better due to its greater impact on server energy. At high temperatures GLB with cluster shutdown runs active clusters at lower utilization to reduce cooling energy. The limited ability of GLB to shutdown clusters at higher temperatures implies that it performs worse than LLB. Thus, GLB outperforms LLB at moderate temperatures outside of these two extremes. The relative performance of GLB versus LLB also depends on the CDN utilization. Figure 10b shows that when the CDN is lightly loaded, GLB has greater flexibility to move traffic around and switch off clusters. There are fewer such opportunities at higher system utilization, where larger clusters need to be kept active for serving the incoming CDN load. At $85°F$, GLB out performs LLB in all cases. But the additional energy savings drop from 42% to 4% as CDN utilization increases from 7% to 35%. This trend is exaggerated when the temperature increases to $100°F$. In this case, LLB is better than GLB but the additional savings provided by LLB increases from 9% to 68% over the same range of utilization.



(a) GLB is better within a broad temperature range

(b) GLB is better at lower utilization and outside temperatures

Figure 10: GLB (cluster shutdown) vs LLB (server shutdown)

5.12. Integrating Server shutdown with Cluster shutdown

We evaluate the hierarchical strategy described earlier in Section 4 that incorporates energy-awareness at both the local and global load balancer by implementing cluster shutdown and server shutdown. A pure cluster shutdown strategy is taken as the baseline, and we study the incremental benefit of adding server shutdown.

We saw earlier in Section 5.7 that with no increase in bandwidth costs ($r = 0$), a pure cluster shutdown strategy kept more clusters active with servers running below the allowable peak utilization ($\mu_{max} = 75\%$). Relaxing bandwidth constraints allowed servers to run at higher utilizations and thus keeping a smaller fraction of its clusters active. In fact, the CDN approached power proportionality for $r = 100\%$. To study the impact of adding server shutdown, we plot the incremental gains obtained in Figure 11a. With no increase in bandwidth cost (r = 0), the combined strategy saves 34% over pure cluster shutdown. Relaxing bandwidth constraints causes savings to drop to a negligible 0.72% at twice the bandwidth cost (r = 100%).

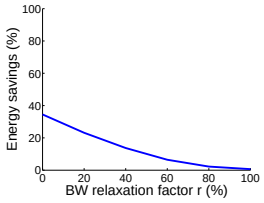
Figure 11b shows incremental gains obtained as a function of performance. If low latency is required, the energy savings over a pure cluster shutdown strategy is 46%, with an average user-cluster distance of 300 km. These gains taper off as performance constraints are relaxed and cluster shutdown approaches power proportionality.

Tight constraints limit the performance of the pure cluster shutdown strategy by requiring the CDN to keep more clusters active and run at higher idle capacity. Server shutdown targets this idle capacity to obtain additional gains. We quantify this in Figure 11c by plotting savings against average idle capacity of an active server (as a percentage of peak utilization μ_{max}). The roughly power proportional nature of the CDN after adding server shutdown implies that any idle capacity previously present is converted directly into savings. This explains the approximate linear nature of the graph.

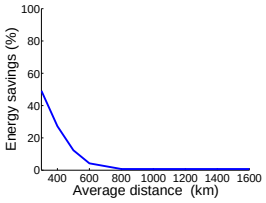
6. Related Work

Data center energy management has emerged as an active area of research in recent years. Several approaches have emerged for reducing the energy consumption of data centers, including server shutdown during off-peak periods [17, 18, 19, 20], the use of low-power server nodes [4], OS-level energy management through methods such as DVFS, the use of renewable energy [21, 22], and routing requests to locations with the cheapest or greener energy [15]. Separately, there has also been work on designing cooling-aware or thermal-aware algorithms for data centers. Cooling-aware workload management techniques have been studied in [23]. Thermal-aware workload placement techniques that place load on cool portions of the data center have been studied in [24, 25]. Models for air- or chiller-based cooling data centers have been studied in [23, 12]; the cooling models used in our paper are inspired by this work and also the data published by the California Energy Commission [14].

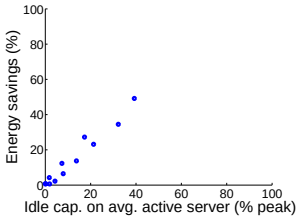
A key difference between the prior work and our work is our focus on content delivery networks; the design choices made by a CDN require these ideas to be customized to the CDN case, for instance by integrating energy management with the CDN's load balancing algorithms. Another key CDN-specific



(a) Additional energy savings over pure cluster shutdown falls off as bandwidth constraints are relaxed. 34% savings are achieved without any increase in BW costs



(b) 46% additional energy savings over pure cluster shutdown can be achieved at an average distance of D = 300 km.



(c) Energy savings obtained by adding server shutdown are roughly linear to the idle capacity of an active server under pure cluster shutdown

Figure 11: Integrating server shutdown with cluster shutdown

issue is to design energy saving methods that minimize the impact on user performance and bandwidth costs. Specifically we use realistic power and cooling models for clusters, based on prior work, and use them to design cluster shutdown algorithms that can be implemented in the CDN's global load balancing algorithms. In this sense the approach also differs from, and is complementary to, prior work on server shutdown technique for CDN energy management [9].

7. Conclusions

We focused on the design of energy-efficient CDNs. Since a CDN could comprise thousands of server clusters across the globe consuming a significant amount of energy, we propose a new technique called cluster shutdown to turn off entire clusters to save energy. Our experimental results using extensive traces from a commercial CDN shows that cluster shutdown can reduce system-wide energy usage by 67% in the optimal case, and most of these savings can be achieved without sacrificing end-user performance and bandwidth costs. In addition, the technique works well even when shutdown is limited to once per day for each cluster and when the load is not known in real-time and must be predicted. We believe that cluster shutdown is a strong candidate for implementation in an actual CDN, especially since it fits in more easily with current CDN architectural principles in comparison with server shutdown techniques studied in the past.

References

[1] L. Barroso, U. Holzle, The case for energy-proportional computing, Computer 40 (12) (2007) 33–37.

[2] J. Koomey, Worldwide electricity used in data centers, Environmental Research Letters 3.

[3] E. Nygren, R. Sitaraman, J. Sun, The Akamai Network: A platform for high-performance Internet applications, ACM SIGOPS Operating Systems Review 44 (3) (2010) 2–19.

[4] D. Anderson, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, V. Vasudevan, Fawn: A fast array of wimpy nodes, in: Proceedings of ACM SOSP, 2009.

[5] M. Weiser, B. Welch, A. Demers, S. Shenker, Scheduling for reduced cpu energy, Mobile Computing (1996) 449–471.

[6] A. Wierman, L. Andrew, A. Tang, Power-aware speed scaling in processor sharing systems, in: INFOCOM 2009, IEEE, IEEE, 2009, pp. 2007–2015.

[7] H. Coles, S. Greenberg, C. Vita, Demonstration of intelligent control and fan improvements in computer room air handlers, Tech. Rep. LBNL-6007E, Lawrence Berkeley National Laboratory (November 2012).

[8] Lawrence Berkeley National Laboratory, Data Center Airflow Management Retrofit, `http://hightech.lbl.gov/documents/data_centers/airflow-doe-femp.pdf` (September 2010).

[9] V. Mathew, R. K. Sitaraman, P. Shenoy, Energy-aware load balancing in content delivery networks, in: INFOCOM, 2012 Proceedings IEEE, IEEE, 2012, pp. 954–962.

[10] J. Rath, DCK Guide To Modular Data Centers: Why Modular?, `http://www.datacenterknowledge.com/archives/2011/10/20/dck-guide-to-modular-data-centers-why-modular/` (October 2011).

[11] B. Pitchaikani, Strategies for the Containerized Data Center, `http://www.datacenterknowledge.com/archives/2011/09/08/strategies-for-the-containerized-data-center/` (September 2011).

[12] S. Pelley, D. Meisner, T. F. Wenisch, J. W. VanGilder, Understanding and abstracting total data center power, in: Proc. of ISCA 2009 Workshop on Energy Efficient Design (WEED), 2009.

[13] M. Stansberry, J. Kudritzki, Uptime Institute 2012 Data Center Industry Survey, Uptime Institute, `http://www.uptimeinstitute.com/images/stories/Uptime_Institute_2012_Data_Industry_Survey.pdf` (2012).

[14] California Energy Commission, Nonresidential Alternative Calculation Method (ACM) approval manual for the 2008 building energy efficiency standards (December 2008).

[15] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, B. Maggs, Cutting the electric bill for internet-scale systems, in: Proceedings of the ACM SIGCOMM 2009 conference on Data communication, ACM, 2009, pp. 123–134.

[16] M. Adler, R. K. Sitaraman, H. Venkataramani, Algorithms for optimizing the bandwidth cost of content delivery, Computer Networks 55 (18) (2011) 4007–4020.

[17] J. Chase, D. Anderson, P. Thakar, A. Vahdat, R. Doyle, Managing energy and server resources in hosting centers, in: Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles (SOSP), 2001, pp. 103–116.

[18] N. Tolia, Z. Wang, M. Marwah, C. Bash, P. Ranganathan, X. Zhu, Delivering energy proportionality with non energy-proportional systems-optimizing the ensemble, in: Proc of Workshop on Power-aware Computing Systems, San Diego, CA, 2008.

[19] A. Krioukov, P. Mohan, S. Alspaugh, L. Keys, D. Culler, R. Katz, Napsac: Design and implementation of a power-proportional web cluster, in: Proc. of ACM Sigcomm workshop on Green Networking, 2010.

[20] A. Chen, W. Das, A. Qin, A. Sivasubramaniam, Q. Wang, N. Gautam, Managing server energy and operational costs in hosting centers, in: Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, 2005.

[21] N. Sharma, S. K. Barker, D. E. Irwin, P. J. Shenoy, Blink: managing server clusters on intermittent power, in: ASPLOS, 2011, pp. 185–198.

[22] I. Goiri, W. Katsak, K. Le, T. Nguyen, R. Bianchini, Parasol and greenswitch: Managing datacenters powered by renewable energy, in: Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2013.

[23] Z. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, M. M. Z. Wang, C. Hyser, Renewable and cooling aware workload management for sustainable data centers, in: Proceedings of ACM Sigmetrics, 2012.

[24] J. Moore, J. Chase, P. Ranganathan, Making scheduling "cool": Temperature-aware workload placement in data centers, in: Proc. USENIX ATC (USENIX '05), 2005.

[25] N. Tolia, Z. Wang, P. Ranganathan, C. Bash, M. Marwah, X. Zhu, Unified thermal and power management in server enclosures, in: Proceedings of the ASME/Pacific Rim Technical Conference and Exhibition (InterPACK '09), 2009.