Managing the Cost, Energy Consumption, and Carbon Footprint of Internet Services [*]

Kien Le[†], Ozlem Bilgir[‡], Ricardo Bianchini[†], Margaret Martonosi[‡], and Thu D. Nguyen[†]

[†]Rutgers University
{lekien,ricardob,tdnguyen}@cs.rutgers.edu

[‡]Princeton University
{obilgir,mrm}@princeton.edu

Technical Report DCS–TR–639, Dept. of Computer Science, Rutgers University, July 2008, Revised December 2009

Abstract

The large amount of energy consumed by Internet services represents significant and fast-growing financial and environmental costs. Increasingly, services are exploring dynamic methods to minimize energy costs while respecting their service-level agreements (SLAs). Furthermore, it will soon be important for these services to manage their usage of "brown energy" (produced via carbon-intensive means) relative to renewable or "green" energy. This paper introduces a general, optimization-based framework for enabling multi-data-center services to manage their brown energy consumption and leverage green energy, while respecting their SLAs and minimizing energy costs. Based on the framework, we propose policies for request distribution across the data centers. Our policies can be used to abide by caps on brown energy consumption, such as those that might arise from Kyoto-style carbon limits, from corporate pledges on carbon-neutrality, or from limits imposed on services to encourage brown energy conservation. We evaluate our framework and policies extensively through simulations and real experiments. Our results show how our policies allow a service to trade off consumption and cost. For example, using our policies, the service can reduce brown energy consumption by 24% for only a 10% increase in cost, while still abiding by SLAs.

1 Introduction

Motivation. Data centers are major energy consumers [16, 2]. In 2006, the data centers in the US consumed 61.4 Billion KWhs, an amount of energy that is equivalent to that consumed by the entire transportation manufacturing industry (the industry that makes automobiles, airplanes, ships, trucks, and other means of transportation). Worse, under current efficiency trends, this gigantic amount of energy will have nearly doubled by 2011 for an overall electricity cost of $7.4 Billion per year [2]. These enormous electricity consumptions translate into large carbon footprints, since most of the electricity produced in the US (and in most other countries) comes from burning coal, a carbon-intensive approach to energy production [26]. (We refer to energy produced by carbon-intensive means as "brown" energy, in contrast with "green" or renewable energy.)

We argue that placing caps on the brown energy consumption of data centers can help businesses, power utilities, and society deal with these challenges. The caps may be government-mandated, utility-imposed, or voluntary. Governments may impose Kyoto-style *cap-and-trade* on large brown energy consumers to curb carbon emissions, encourage energy conservation, and promote green energy. For example, the UK government will start a mandatory cap-and-trade scheme for businesses consuming more than 6 GWh per year in April 2010 [36]; i.e., a business with even a relatively small 700-KW data center will have to participate. If its brown cap is exhausted, the business will have to purchase offsets from the market. Congress is now discussing a federal cap-and-trade scheme for the US, while regional schemes have already been created [33].

Utilities may impose caps on large electricity consumers to encourage energy conservation or manage their own costs. In this scenario, consumers that exceed the cap could pay higher brown electricity prices, in a scheme we call *cap-and-pay*.

Finally, businesses may voluntarily set brown energy caps for themselves in an effort to manage their energy-related costs; in this scenario, caps translate into explicit targets for energy conservation. When carbon-neutrality can be used as a marketing tool, businesses may also use caps to predict their expenditures with neutrality and/or green energy. We refer to these voluntary caps as *cap-as-target*.

This paper. Regardless of the capping scheme, *the research question is how to create the software support for capping brown energy consumption*[1] *without excessively increasing costs or degrading performance. This question has not been addressed before, since we are the first to propose and evaluate approaches to cap brown energy consumption.*

In this paper, we seek to answer this question in the context of Internet services. These services are supported by multiple data centers for high capacity and availability, and low response times. The data centers sit behind front-end devices that inspect each client request and forward it to one of the data centers that can serve it, according to a *request distribution policy*. Despite their wide-area distribution of requests, services must strive not to violate their service-level agreements (SLAs).

[*]This research was partially supported by NSF grants #CSR-0916518 and #CNS-0448070, and Rutgers grant #CCC-235908.

[1]The energy caps we study should not be confused with power caps, which are used to limit the "instantaneous" power draw of data centers.

Specifically, we propose and evaluate a software framework for optimization-based request distribution. The framework enables services to manage their energy consumption and costs, while respecting their SLAs. For example, the framework considers the energy cost of processing requests before and after the brown energy cap is exhausted. Under cap-and-trade, this involves tracking the market price of carbon offsets and interacting with the market after cap exhaustion. At the same time, the framework considers the existing requirements for high throughput and availability. Furthermore, the framework allows services to exploit data centers that pay different (and perhaps variable) electricity prices, data centers located in different time zones, and data centers that can consume green energy (either because they are located close to green energy plants or because their power utilities allow them to select a mix of brown and green energy, as many utilities do today). Importantly, the framework is general enough to enable energy management in the absence of brown energy caps, different electricity prices, or green energy.

Based on the framework, we propose request distribution policies for cap-and-trade and cap-and-pay. Operationally, an optimization-based policy defines the fraction of the clients' requests that should be directed to each data center. The front-ends periodically (e.g., once per hour) solve the optimization problem defined by the policy, using mathematical optimization algorithms, time series analysis for load prediction, and statistical performance data from data centers. After fractions are computed, the front-ends abide by them until they are recomputed. For comparison, we also propose a simpler heuristic policy that is greedy and operates quite differently. During each hour, it first exploits the data centers with the best power efficiency, and then starts exploiting the data centers with the cheapest electricity.

Using simulation, a request trace from a commercial service, and real network latency, electricity price, and carbon market traces, we evaluate our optimization and heuristic-based request distributions in terms of energy costs and brown energy consumptions. We also investigate the impact of the capping scheme, the size of the cap, the performance of the data centers, and server energy proportionality. Using a real system distributed over four universities, we validate the simulation with real experiments.

We make several interesting observations from our results. First, our optimization-based distributions achieve lower energy costs than our simpler heuristic. Our best optimization-based distribution achieves 35% lower costs (and similar brown energy consumption) while meeting the same SLA. Second, we demonstrate that predicting load intensities many hours into the future reduces costs significantly. Third, we show that brown energy caps and data centers that exploit green energy can be used to limit brown energy consumption without significantly increasing energy costs or violating the SLA. For example, we can reduce the brown energy consumption by 24% for a 10% increase in cost. Fourth, we find that our optimization-based distribution can achieve cost reductions of 24% by exploiting different electricity prices at widely distributed data centers, even in the absence of brown energy caps and green energy.

Contributions. The vast majority of the previous work on data center energy management has focused on a single data center. Furthermore, no previous work has considered the problem of capping the brown energy consumption of Internet services or interacting with the carbon market. Thus, this paper makes the following contributions: (1) we propose a general, optimization-based framework for minimizing the energy cost of services in the presence of brown energy caps, data centers that exploit green energy, data centers that pay different electricity prices, data centers located in different time zones, and/or the carbon market; (2) based on the framework, we propose request distribution policies for minimizing the energy cost while abiding by SLAs; (3) we propose a simpler, heuristic policy for the same purpose; and (4) we evaluate our framework and policies extensively through simulation and experimentation.

Roadmap. The next section describes the background behind our work. Section 3 describes our framework and its solution approaches, as well as our heuristic policy. Section 4 describes our methodology and results. Section 5 overviews the related work. Finally, Section 6 draws our conclusions.

2 Background

Current request distribution policies. When multiple data centers are capable of serving a collection of requests, i.e. they are mirrors with respect to the content requested, the service's front-end devices can intelligently distribute the offered request load. Typically, a request can only be served by 2 or 3 mirrors; further replicating content would increase state-coherence traffic without a commensurate benefit in availability or performance.

Current request-distribution policies over the wide area typically attempt to balance the load across mirror data centers, minimize the response time seen by clients, and/or guarantee high availability, e.g. [5, 31, 37]. For example, round-robin DNS [5] attempts to balance the load by returning the address of a different front-end for each DNS translation request. More interestingly, Ranjan *et al.* [31] redirect dynamic-content requests from an overloaded data center to a less loaded one, if doing so is likely to produce a lower response time.

Carbon market dynamics. In cap-and-trade scenarios, our policies make decisions based in part on the market price of carbon offsets. To illustrate how this market behaves and the feasibility of basing decisions on it, Figures 1–3 show the price (in Euros) of 1 ton of carbon in the futures market for December 2008 at different time-granularities. Figure 1 depicts how the December 2008 futures price fluctuated on a single day: April 3, 2008. Figure 2 depicts the December 2008 futures price as collected during the entire week of March 31, 2008. Figure 3 depicts the December 2008 futures price as collected during the month of December 2007. The data plotted in the figures was obtained from [25].

Although there can be periods of variability, these figures show that the overall increasing or decreasing price trends remain clear, even at relatively short time scales. In addition, although the graphs span relatively short periods, they show exploitable price variations—on the order of 10%. Finally, we can see that it typically takes at least a few hours for prices to change appreciably.

These observations suggest that the market has enough variability to make dynamic decisions worthwhile. Our framework and policies enable services to take full advantage of price variations. Moreover, the market's stability suggests that market-based decisions will be meaningful; excessive variability or the absence of clear trends could render our decisions inadequate in just a short time. Nevertheless, if a period of instability is ever detected, our policies can be tuned to mitigate the impact of the instability on the request distribution.

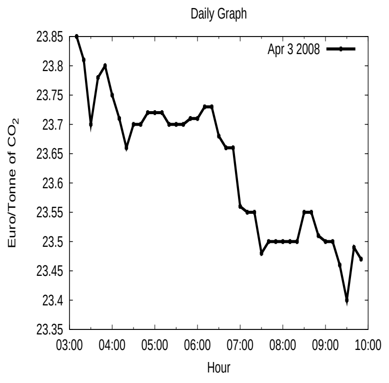

Figure 1: Prices on April 3, 2008. Decreasing trend.

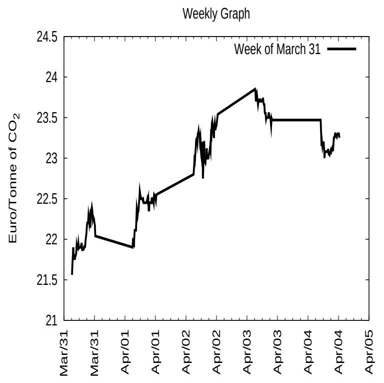

Figure 2: Prices on week of March 31, 2008. Increasing trend.

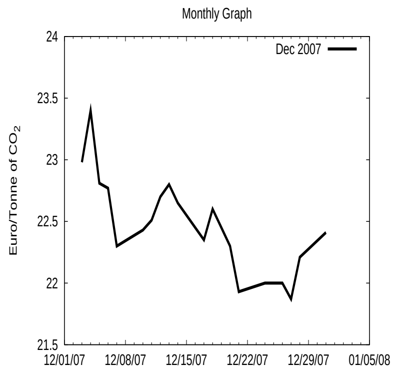

Figure 3: Prices in Dec. 2007. Decreasing trend.

3 Request Distribution Policies

Our ultimate goal is to design request-distribution policies that minimize the energy cost of a multi-data-center Internet service, while meeting its SLA. Next, we discuss the principles and guidelines behind our policies, and then present each policy in turn.

3.1 Principles and Guidelines

For our policies to be practical, it is not enough to minimize energy costs; we must also guarantee high performance and availability, and do it all dynamically. Our policies respect these requirements by having the front-ends (1) prevent data center overloads; and (2) monitor their response times, and adjust the request distribution to correct any performance or availability problems.

When data centers can use green energy, we assume that the power mix for each center is contracted with the corresponding power utility for an entire year. The information used to determine the mixes comes from the previous year. The brown energy cap is associated with the entire service (i.e., all of its data centers) and also corresponds to a year, the "energy accounting period". Any leftover brown energy can be used in the following year. When a service exceeds the cap, it must either purchase carbon offsets corresponding to its excess consumption (cap-and-trade) or start paying higher electricity prices (cap-and-pay). The service has a single SLA with customers, which is enforced on a weekly basis, the "performance accounting period". The SLA is specified as (L, P), meaning that at least P% of the requests must complete in less than L time, *as observed by the front-ends*. This definition means that any latency added by the front-ends in distributing requests to distant data centers is taken into account. Our SLA can be combined with Internet QoS approaches to extend the guarantees all the way to the users' sites [39].

Our SLA implies that *the service does not need to select a front-end device and data center that are closest to each client for the lowest response time possible*; all it needs is to have respected the SLA at the end of each week. However, some services may not be able to tolerate increases in network latency. For those services, set L low and P high. Other services are more flexible. For example, many services have heavy processing requirements at the data centers themselves; additional network latency would represent a small fraction of the overall response time. For those services, L can be increased enough to allow more flexibility in the request distribution and lower energy costs. In Section 4, we investigate the tradeoff between the response time requirements (defined by the SLA) and our ability to minimize costs.

We assume that each data center reconfigures itself by leaving only as many servers active as necessary to service the expected load for the next hour plus an additional 20% slack. (The extra servers can deal with unexpected increases in load and compensate for any inaccuracy of our load prediction. In fact, as we shall demonstrate in Section 4, our predictions in a few cases underestimate the load by roughly 10%. The 20% slack includes an additional safety margin of 10% on top of this small prediction inaccuracy.) The other servers can be turned off to conserve energy, as in [6, 7, 8, 14, 24]. Turning servers off does not affect the service's data set, as we assume that servers rely on network-attached storage (and local Flash), like others have done as well, e.g. [6]. For such servers, the turn-on delay is small and can be hidden by the slack. In addition, their turn-on/off energy is negligible compared to the overall energy consumed by the service (transitions occur infrequently, as shall also be seen in Section 4). The reason is that turning a server on takes on the order of one minute [8, 24] (turning it off is substantially faster), whereas the minimum reconfiguration interval in our systems is one hour. For this reason, we do not model the transition energy explicitly throughout the paper.

3.2 Optimization-Based Distribution

Our framework comprises the parameters listed in Table 1. Using these parameters, we can formulate optimization problems defining the behavior of our request distribution policies. The optimization seeks to define the power mixes m_i^{brown},m_i^{green} for each data center i, once per year. At a finer granularity, the optimization also seeks to define the fraction $f_i^y(t)$ of requests of type y that should be sent by the front-end devices to each data center i, during "epoch" t. During each epoch, the fractions are fixed. Depending on the solution approach, a set of fractions is computed for one or more epochs at a time. *The computation is performed off the critical path of request distribution.* The front-ends must recompute the fractions, if the load intensity, the electricity price, the data center response times, or the market prices change significantly since the last computation. A solution is typically computed no more than twice per hour.

After each recomputation and/or every hour, the front-ends inform the data centers about their expected loads for the next hour.

Symbol	Meaning
$f_i^y(t)$	Percentage of requests of type y to be forwarded to center i in epoch t
m_i^{brown}	Percentage of brown electricity in the power mix of center i
m_i^{green}	Percentage of green electricity in the power mix of center i
Overall Cost	Total energy cost (in \$) of the service
BEC	Brown energy cap (in KWh) of all centers combined
$p^y(t)$	Percentage of requests of type y in the workload in epoch t
M_i^y	Equals 1 if data center i can serve requests of type y. Equals 0 otherwise
$ReqCost_i^y(t)$	Avg. energy cost (in \$) incl. cap-violation charges of serving a request of type y at center i in epoch t
$BaseCost_i(\mathit{offered}_i, t)$	Base energy cost (in \$) incl. cap-violation charges of center i in epoch t, under $\mathit{offered}_i$ load
$b_i^y(t)$	Avg. energy cost (in \$) of serving a request of type y using only brown electricity at center i
$g_i^y(t)$	Avg. energy cost (in \$) of serving a request of type y using only green electricity at center i
$b_i^{base}(\mathit{offered}_i, t)$	Base energy cost (in \$) of center i in epoch t, under $\mathit{offered}_i$ load, using only brown electricity
$g_i^{base}(\mathit{offered}_i, t)$	Base energy cost (in \$) of center i in epoch t, under $\mathit{offered}_i$ load, using only green electricity
$\mathit{market}_i^y(t)$	Avg. carbon market cost (in \$) to offset a request of type y in center i in epoch t
$\mathit{marketBase}_i(\mathit{offered}_i, t)$	Market cost (in \$) of offsetting the base energy in center i in epoch t, under $\mathit{offered}_i$ load
$\mathit{fee}_i^y(t)$	Avg. fee for cap violation (in \$) for a request of type y in center i in epoch t
$\mathit{feeBase}_i(\mathit{offered}_i, t)$	Fee for cap violation (in \$) for the base energy in center i in epoch t, under $\mathit{offered}_i$ load
LR(t)	Expected peak service load rate (in reqs/sec) for epoch t
LT(t)	Expected total service load (in number of reqs) for epoch t
LC_i	Load capacity (in reqs/sec) of center i
$CDF_i(L, \mathit{offered}_i)$	Expected percentage of requests sent to center i that complete within L time, given $\mathit{offered}_i$ load
$\mathit{offered}_i$	$\sum_y f_i^y(t) \times M_i^y \times p^y(t) \times LR(t)$ (in reqs/sec)

Table 1: Framework parameters. Note that we define the "peak" service load rate, $LR(t)$, as the 90th percentile of the actual load rate to eliminate outlier load rates. Also, note that $b_i^y(t)$, $g_i^y(t)$, $market_i^y(t)$, and $fee_i^y(t)$ exclude the cost of the "base" energy.

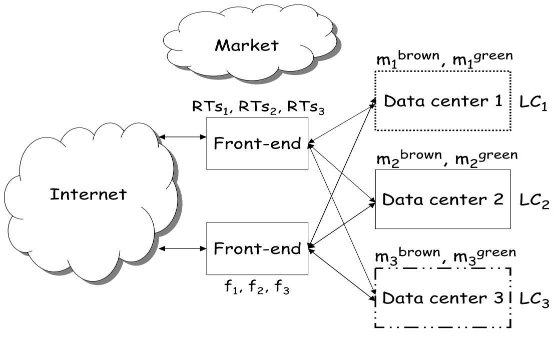
Market
m_1^{brown}, m_1^{green}
RTs_1, RTs_2, RTs_3
Data center 1
LC_1
Front-end
m_2^{brown}, m_2^{green}
Internet
Data center 2
LC_2
Front-end
m_3^{brown}, m_3^{green}
Data center 3
LC_3
f_1, f_2, f_3

Figure 4: Each data center i consumes m_i^{brown}% brown energy and m_i^{green}% green energy, and has a certain processing capacity (LC) and a history of response times (RTs). The f_i's are the percentages of requests that should be sent to the data centers.

With this information, the data centers can reconfigure by leaving only as many servers active as necessary to service the expected load (plus the 20% slack). Figure 4 depicts an example service and the main characteristics of its data centers.

The next subsection describes two specific optimization problems (policies), one for cap-and-trade and one for cap-and-pay. Subsection 3.2.2 describes the instantiation of the parameters. Subsection 3.2.3 discusses how to solve the problems.

3.2.1 Problem Formulations

Cap-and-trade policy. Our first optimization problem seeks to optimize the overall energy cost, *Overall Cost*, in a cap-and-trade scenario. Equation 1 of Figure 5 defines *Overall Cost*, where $p^y(t)$ is the percentage of requests of type y in the service's workload during epoch t, $LT(t)$ is the expected total number of requests for the service during epoch t, $ReqCost_i^y(t)$ is the average energy cost (including cap-violation charges) of processing a request of type y at data center i during epoch t, and $BaseCost_i(\mathit{offered}_i, t)$ is the "base" energy cost (including any cap-violation charges) of data center i during epoch t, under a given $\mathit{offered}_i$ load. Base energy is the energy that is spent when the active servers are idle; it is zero for a perfectly energy-proportional system [3], and non-zero for systems with idle power dissipation. We define $\mathit{offered}_i$ as $\sum_y f_i^y(t) \times M_i^y \times p^y(t) \times LR(t)$, where M_i^y is set when center i can serve requests of type y and $LR(t)$ is the peak request rate during epoch t.

The per-request cost, $ReqCost_i^y(t)$, is defined in Equation 2, where $g_i^y(t)$ is the average cost of serving a request of type y at center i using only green electricity in epoch t, $b_i^y(t)$ is the same cost when only brown electricity is used, BEC is the brown energy cap for the service, and $market_i^y(t)$ is the average cost of carbon offsets equivalent to the brown energy consumed by center i on a request of type y in epoch t.

Essentially, this cost-per-request model says that, before the brown energy cap is exhausted, the cost of executing an average request of a certain type is the average (weighted by the power mix) of what it would cost using completely green or brown electricity. Beyond the cap exhaustion point, the service needs to absorb the additional cost of purchasing carbon offsets on the market. This cost model also means that *the optimization problem is non-linear when the power mixes have not yet been computed* (since the mixes are multiplied by the fractions in *Overall Cost*).

The base energy cost $BaseCost_i(\mathit{offered}_i, t)$ is defined similarly in Equation 3. In this equation, $b_i^{base}(\mathit{offered}_i, t)$ is the base energy cost of center i in epoch t under brown electricity prices, $g_i^{base}(\mathit{offered}_i, t)$ is the same cost but under green electricity prices, and $marketBase_i(\mathit{offered}_i, t)$ is the market cost of offsetting the base energy of center i in epoch t.

Overall Cost should be minimized under the constraints that

$$\textit{Overall Cost} = (\sum_t \sum_i \sum_y f_i^y(t) \times M_i^y \times p^y(t) \times LT(t) \times \textit{ReqCost}_i^y(t)) + (\sum_t \sum_i BaseCost_i(\textit{offered}_i, t)) \tag{1}$$

$$ReqCost_i^y(t) = \begin{cases} m_i^{brown} \times b_i^y(t) + m_i^{green} \times g_i^y(t), \textit{if brown energy consumed so far } \leq BEC \\ market_i^y(t) + m_i^{brown} \times b_i^y(t) + m_i^{green} \times g_i^y(t), \textit{otherwise} \end{cases} \tag{2}$$

$$BaseCost_i(\textit{offered}_i, t) = \begin{cases} m_i^{brown} \times b_i^{base}(\textit{offered}_i, t) + m_i^{green} \times g_i^{base}(\textit{offered}_i, t), \textit{if brown energy consumed so far } \leq BEC \\ marketBase_i(\textit{offered}_i, t) + m_i^{brown} \times b_i^{base}(\textit{offered}_i, t) + m_i^{green} \times g_i^{base}(\textit{offered}_i, t), \textit{otherwise} \end{cases} \tag{3}$$

1. $\forall i\ m_i^{brown}$ *and* $m_i^{green} \geq 0 \Rightarrow$ *i.e., each part of the mix cannot be negative.*
2. $\forall i\ m_i^{brown} + m_i^{green} = 1 \Rightarrow$ *i.e., the mixes need to add up to 1.*
3. $\forall t \forall i \forall y\ f_i^y(t) \geq 0 \Rightarrow$ *i.e., each fraction cannot be negative.*
4. $\forall t \forall y \sum_i f_i^y(t) \times M_i^y = 1 \Rightarrow$ *i.e., the fractions for each request type need to add up to 1.* (4)
5. $\forall t \forall i\ \textit{offered}_i \leq LC_i \Rightarrow$ *i.e., the offered load to a data center should not overload it.*
6. $\sum_t \sum_i \sum_y (f_i^y(t) \times M_i^y \times p^y(t) \times LT(t) \times CDF_i(L, \textit{offered}_i)) \,/\, \sum_t LT(t) \geq P \Rightarrow$ *i.e., the SLA must be satisfied.*

Figure 5: Cap-and-trade formulation. We solve our formulations to find $f_i^y(t)$, m_i^{brown}, and m_i^{green}. The goal is to minimize the energy cost (*Overall Cost*), under the constraints that no data center should be overloaded and the SLA should be met. The energy cost is the sum of dynamic request-processing (*ReqCost*) and static (*BaseCost*) costs, including any carbon market costs due to violating the brown energy cap.

follow the equations above, where LC_i is the processing capacity of center i, and $CDF_i(L, \textit{offered}_i)$ is the percentage of requests that were served by center i within L time (as observed by the front-ends) when it most recently received *offered*$_i$ load. Figure 5 includes summary descriptions of the constraints. *Note that we could have easily added a constraint to limit the distance between a front-end and the centers to which it can forward requests. This would provide stricter limits on response time than our SLA.*

After the optimization problem is solved once to compute the power mixes for the year and the first set of fractions, the power mix variables can be made constants and the power mix-related constraints can be disregarded. Under these conditions, the problem can be made linear by solving it for one epoch at a time.

Cap-and-pay policy. The formulation just presented assumes that services may purchase carbon offsets on an open trading market to compensate for having exceeded their brown energy caps. If carbon offsets happen to be cheap, the service does not have a significant incentive to conserve as much brown energy as possible. If governments or power utilities decide that they want to promote brown energy conservation (and green energy production/consumption), an alternative is to levy high fees on the service when brown energy caps are exceeded. We also study this scenario. The only modification required to the formulation above is to replace $market_i^y(t)$ and $marketBase_i(\textit{offered}_i, t)$ with $fee_i^y(t)$ and $feeBase_i(\textit{offered}_i, t)$, respectively.

Cap-as-target. A cap-as-target formulation would be similar to those above. The only difference would be the penalty for violating the self-imposed cap, which would account for the additional costs of achieving carbon neutrality (e.g., the cost of planting trees). As cap-and-trade and cap-and-pay are enough to explore the benefits of our framework and policies, we do not consider cap-as-target further in this paper.

Applying our formulations to today's services. It is important to mention that *the formulations mentioned above are useful even for current Internet services, i.e. those without brown energy caps or green energy consumption.* The brown data centers that support these services are spread around the country (and perhaps even around the world) and are exposed to different brown electricity prices. In this scenario, our framework and policies can optimize costs while abiding by SLAs, by leveraging the different time zones and electricity prices. To model these services, all we need to do is specify "infinite" energy caps and 100%–0% power mixes for all data centers.

Complete framework and policies. For clarity, the description above did not address services with session state (i.e., soft state that only lasts the user's session with the service) and on-line writes to persistent state (i.e., writes are assumed to occur out-of-band). Sessions need not be explicitly considered in the framework or policy formulations, since services ultimately execute the requests that form the (logical) sessions. (Obviously, sessions must be considered when actually distributing requests, since multiple requests belonging to the same session should be executed at the same data center.) Dealing with on-line writes to persistent state mostly requires extensions to account for the energy consumed in data coherence activities. We present the extended framework and a cap-and-trade formulation including writes in Appendix A. In addition, we briefly evaluate the effect of session length and percentage of write requests in Section 4.

3.2.2 Instantiating Parameters

Before we can solve our optimization problems, we must determine input values for their parameters. However, doing so is not straightforward in the presence of multiple front-end devices. For example, as aforementioned, the response time of the data centers must be measured at each front-end device. Moreover, no front-end sees the entire load offered to the service. Thus, to select the parameters exactly, the front-ends would have to communicate and coordinate their decisions. To avoid these overheads, we explore a simpler approach in which the optimization problem is solved independently by each of the front-ends. If the front-ends guarantee that the constraints are satisfied from their independent points of view, the constraints will be satisfied globally.

In this approach, $LT(t)$ and $LR(t)$ (and consequently *offered*$_i$) are defined for each front-end. In addition, the load capacity of each data center is divided by the number of front-ends. To instantiate CDF_i, each front-end collects the recent history of response times of data center i when the front-end directs *offered*$_i$ load to it. For this purpose, each front-end has a table of these <offered load, percentage of requests served within L time> entries for each data center that is filled over time. Each table has

4 entries corresponding to different levels of utilization (up to 25%, between 25% and 50%, between 50% and 75%, and between 75% and 100%). Similarly, we create a table of <offered load, base energy consumption> entries for each data center. The entries are filled by computing the ratio of the peak load and the load capacity of the data center, and assuming that the same ratio of the total set of servers (plus the 20% slack) is left active. This table only needs to be re-instantiated when servers are upgraded or added to/removed from the data center.

Because the CDF_i and base-energy tables include entries covering the entire range of *offered*$_i$ and consequently $f_i^y(t)$, a solution to our optimization problem will actually consider the effect of any $f_i^y(t)$ we may select on the performance and energy cost of data center i. This aspect of our formulation contributes heavily to the stability of our request distribution approach.

Whenever a solution to the problem needs to be computed, the only runtime information that the front-ends need from the data centers is the amount of energy that they have already consumed during the current energy accounting period. Other information, such as load capacities and electricity costs, can be readily available to all front-ends. In our future work, we will study the tradeoff between the overhead of direct front-end communication in solving the problem and the quality of the solutions.

3.2.3 Solution Approaches

We study three solution approaches that differ in the extent to which they use load-intensity prediction and linear programming (LP) techniques. Next, we discuss each of the approaches in turn. The discussion assumes the carbon-curbing formulation but extends trivially to the other formulations as well.

Simulated Annealing (SA): Solving for mixes and fractions using week-long predictions. The solution of the optimization problem for an entire energy accounting period (one year) provides the best energy cost. However, such a solution is only possible when we can predict future offered load intensities, electricity prices, carbon market prices, and data center response times.

Electricity price predictions are trivial when the price is constant or when there are only two prices (on-peak and off-peak prices). Other predictions are harder to make far into the future. Instead, we predict detailed behaviors for the near future (the next week, matching the performance accounting period) and use aggregate data for the rest of the year. Specifically, our approach divides the brown energy cap into 52 chunks, i.e. one chunk per week. The energy associated with each chunk is weighted by the aggregate amount of service load predicted for the corresponding week. The intuition is that the amount of brown energy required is proportional to the offered load. Based on the chunk defined for the next week, we solve the optimization problem. Thus, we only need detailed predictions for the next week.

For predicting load intensities during this week, we consider Auto-Regressive Integrated Moving Average (ARIMA) modeling [4]. We do not attempt to predict carbon market prices or CDF_i. Instead, we assume the current market price and the current CDF_i tables as predictions. (As explained below, we recompute the request distribution, i.e. the fractions f_i^y, when these assumed values have become inaccurate.)

Unfortunately, in this solution approach we cannot use LP solvers, which are very fast, even when the power mixes have already been computed. The reason is that, when we compute results for many epochs at the same time, we are still left with a few non-linear functions (i.e., $ReqCost_i^y$, the load intensities, and consequently, $BaseCost_i$ and CDF_i). Instead of LP, we use Simulated Annealing [17] and divide the week into 42 4-hour epochs, i.e. $t = 1..42$. (We could have used a finer granularity at the cost of longer processing time.) For each epoch, the load intensity is assumed to be the predicted "peak" load intensity (actually, the 90th-percentile of the load intensity to exclude outlier intensities) during the epoch.

```
Solve the problem to define mixes for the year and
  fractions for every 4-hour epoch of the first week;
Request mixes from utilities;
Every week do
  If !first week then
    Solve the problem to define fractions for every
      4-hour epoch of the week;
  Start distributing requests according to fractions;
  Every epoch of the week do
    Recompute fractions at the end of an epoch if
      any electricity price has changed significantly
      predictions were inaccurate for the epoch
      the predicted load for the next epoch is
        significantly different than the current load;
    Recompute fractions immediately if
      the brown energy cap expires
      a data center becomes unavailable;
    After each recomputation and every hour
      Install new fractions (if they were recomputed);
      Inform data centers about their predicted loads
        for the next hour;
  End do;
End do;
```

Figure 6: Overview of the SA solution approach.

We use this approach to determine the power mixes once per year. After this first solution, we may actually recompute the request distribution in case our predictions become inaccurate over the course of each week. Specifically, at the end of an epoch, we recompute if any electricity price has changed significantly (this may only occur when real-time, hourly electricity pricing is used [19, 27, 28]), or the actual peak load intensity was either significantly higher or lower (by more than 10% in our experiments) than the prediction. We do the same for parameters that we do not predict (market price and CDF_i); at the end of an epoch, we recompute if any of them has changed significantly with respect to its assumed value. We also recompute at the end of an epoch if the predicted load for the next epoch is significantly higher or lower than the current load. In contrast, we recompute immediately whenever the brown cap is exhausted or a data center becomes unavailable, since these are events that lead to substantially different distributions. Given these conditions, recomputations occur at the granularity of many hours in the worst case.

Obviously, we must adjust the brown energy cap every time the problem is solved to account for the energy that has already been consumed. Similarly, we must adjust the SLA percentage P, according to the requests that have already been serviced within L time. Each recomputation produces new results only for the rest of the current week. Any leftover energy at the end of a week is added to the share of the next week.

Finally, after a recomputation occurs and every hour, the front-ends inform the data centers about their predicted loads for the next hour (i.e., the predicted load intensity at each front-end times the fraction of requests to be directed to each data center). Each data center collects the predictions from all front-ends and reconfigures its set of active servers accordingly. Figure 6 summarizes the SA approach to solving our optimization problems.

LP1: Solving for fractions using one-hour predictions. As we

```
Solve the problem using SA to define mixes for the year;
Request mixes from utilities;
Solve the problem using LP to define fractions for the
  first 1-hour epoch;
Start distributing requests according to fractions;
Every epoch do
  Recompute fractions at the end of an epoch if
    any electricity price has changed
      significantly or
    predictions were inaccurate for the epoch or
    the predicted load for next epoch is significantly
      different than the current load;
  Recompute fractions immediately if
    the brown energy cap expires or
    a data center becomes unavailable;
  After each recomputation
    Install new fractions;
    Inform data centers about their new predicted loads;
End do;
```

Figure 7: Overview of the LP1 solution approach.

mentioned above, using week-long predictions means that fast LP solvers cannot be applied. However, if we assume shorter epochs of only 1 hour and focus solely on the next hour ($t = 1$), we can convert the optimization problem into an LP problem and solve it to compute the fractions f_i^y.

The conversion works by transforming non-linear functions into constant values. Specifically, we transform $ReqCost_i^y$ into the expected per-request costs for the next epoch, since we know whether the energy cap has been exceeded. For the load intensity, we predict it to be the expected peak load (as represented by the 90th percentile) rate for the next hour, again using ARIMA modeling. With the expected load, we transform $BaseCost_i$ into the expected base energy costs for the next epoch. Like we did above, we do not predict carbon market prices or CDF_i, instead using their current values as a prediction for the next hour.

We recompute a solution under the same conditions as SA: (1) at the end of an epoch for inaccurate load predictions or assumed values; (2) at the end of an epoch, when a prediction for the next epoch is significantly different than that for the current epoch; and (3) immediately when the brown energy cap expires or a data center becomes unavailable. We adjust the cap as before. Given these conditions, recomputations (which are much faster than those of SA) occur more frequently than under SA, but typically no more frequently than every hour.

After a recomputation, the front-ends inform the data centers about their predicted loads for the next hour. The data centers reconfigure accordingly. Figure 7 summarizes the LP1 approach to solving our optimization problems.

LP0: Solving for fractions without using predictions. Finally, we can solve our optimization problem without using any predictions at all. We do so using 30-minute epochs and focusing solely on the next epoch ($t = 1$) with an LP solver. Every time the problem is solved, we use the current values for all problem parameters. For example, the "current" load offered to a data center is its average offered load in the last 5 minutes. As in SA and LP1, we check for deviations at the end of each epoch and recompute a solution if any significant deviations occur. We also recompute immediately, if the cap is exhausted or a center becomes unavailable. Due to its lack of predictions, we expect this approach to cause recomputations nearly every 30 minutes.

After a recomputation, the front-ends inform the data centers about their expected loads for the next half hour (the current load offered to each front-end times the fraction of requests to be directed to each center). The data centers reconfigure accordingly. Figure 8 summarizes the LP0 approach to solving our optimization problems.

```
Solve the problem using SA to define mixes for the year;
Request mixes from utilities;
Solve the problem using LP to define fractions for the
  first 30-minute epoch;
Start distributing requests according to fractions;
Every epoch do
  Recompute fractions at the end of an epoch if
    the current values for the formulation parameters
      are significantly different than when the
      problem was last solved;
  Recompute fractions immediately if
    the brown energy cap expires or
    a data center becomes unavailable;
  After each recomputation
    Install new fractions;
    Inform data centers about their new expected loads;
End do;
```

Figure 8: Overview of the LP0 solution approach.

3.3 Heuristics-Based Request Distribution

We also propose a cost-aware heuristic policy (CA-Heuristic) that is simpler and less computationally intensive than the optimization-based approaches described above. The policy deals only with dynamic request distribution; the power mixes are still computed using the optimization-based solution with week-long predictions (SA).

CA-Heuristic is greedy and uses 1-hour epochs. It tries to forward each request to the best data center that can serve it (based on a metric described below), without violating two constraints: (1) the load capacity of each data center; and (2) the SLA requirement that P% of the requests complete in less than L time, as seen by the front-end devices. To avoid the need for coordination between front-ends, they divide the load capacity of each data center by the number of front-ends. In addition, each front-end verifies that the SLA is satisfied from its point of view.

The heuristic works as follows. At each epoch boundary, each front-end computes $R = P \times E$ (the number of requests that must have lower latency than L), where E is the number of requests the front-end expects in the next epoch. E can be predicted using ARIMA. Each front-end also orders the data centers that have $CDF_i(L, LC_i) \geq P$ according to the ratio $Cost_i(t)/CDF_i(L, LC_i)$, from lowest to highest ratio, where $Cost_i(t)$ is the average cost of processing a request (weighted across all types) at data center i during epoch t. The remaining data centers are ordered by the same ratio. A final list, called *MainOrder*, is created by concatenating the two lists.

Requests are forwarded to the first data center in MainOrder until its capacity is met. At that point, new requests are forwarded to the next data center on the list and so on. After the front-end has served R requests in less than L time, it can disregard MainOrder and start forwarding requests to the cheapest data center (lowest $Cost_i(t)$) until its capacity is met. At that point, the next cheapest data center can be exercised and so on.

If the prediction of the number of requests to be received in an epoch consistently underestimates the offered load, serving R requests within L time may not be enough to satisfy the SLA. To prevent this situation, whenever the prediction is inaccurate, the heuristic adjusts the R value for the next epoch to compensate.

At each epoch boundary, the front-ends inform the centers

Characteristic	SA	LP1	LP0	CA-Heuristic
Energy accounting period	1 year	1 year	1 year	1 year
Power mix computation	SA once/year	SA once/year	SA once/year	SA once/year
Performance accounting period	1 week	1 week	1 week	1 week
Epoch length	4 hours	1 hour	1/2 hour	1 hour
Load predictions	Per front-end for next week	Per front-end for next epoch	None	Per front-end for next epoch
Recomputation/reordering decision	Epoch boundary	Epoch boundary	Epoch boundary	Epoch boundary
Communication with DCs	Yes	Yes	Yes	Yes

Table 2: Main characteristics of policies and solution approaches.

about their predicted loads for the next epoch. These predictions are based on the per-front-end load predictions and their Main-Order lists (the lists may change because of changes to *Cost*$_i$).

Table 2 overviews our policies and solution approaches.

4 Evaluation

4.1 Methodology

To evaluate our framework and policies, we use both simulation and real-system experimentation. Our simulator of a multi-data-center Internet service takes as input a request trace, an electricity price trace, and a carbon market trace or a fee trace. Using these traces, it simulates a request distribution policy, and the data center response times and energy consumptions. Our evaluations are based on year-long traces, as well as sensitivity studies varying our main simulation parameters.

For simplicity, we simulate a single front-end device located on the East Coast of the US. The front-end distributes requests to 3 data centers, each of them located on the West Coast, on the East Coast, and in Europe. *The simulator has been validated against a real prototype implementation, running on servers at four universities located in these same regions* (University of Washington, Rutgers University, Princeton University, and EPFL). We present our validation results in subsection 4.2.

Request trace, time zones, and response times. Our request trace is built from a 1-month-long real trace from a commercial search engine, Ask.com. The trace corresponds to a fraction of the requests Ask.com received during April 2008. Due to commercial and privacy concerns, the trace only lists the number of requests for each second. (Even though search engines are sensitive to increases in response time, this trace is representative of the traffic patterns of many real services, including those that can tolerate such increases. In our work, the request traffic, and our ability to predict it and intelligently distribute it are much more important than the actual service being provided.)

To extrapolate the trace to an entire year, we use the search volume statistics for Ask.com from Nielsen-online.com. Specifically, we normalize the total number of requests for other months in 2008 to those of April. For example, to generate the trace for May 2008, we multiply the load intensity of every second in April by the normalized load factor for May.

Figure 9 shows the 90th percentile of the actual and ARIMA-predicted request rates during one week of our trace. Our ARIMA modeling combines seasonal and non-seasonal components additively. The non-seasonal component involves 3 auto-regressive parameters and 3 moving average parameters (corresponding to the past three hours). The seasonal component involves 1 auto-regressive parameter and 1 moving average parameter (corresponding to the same hour of the previous week). The figure shows that the ARIMA predictions are very accurate.

For simplicity, we assume that all requests are of the same type and can be sent to any of the 3 data centers. A request takes 400 ms to process on average and consumes 60 J of dynamic energy (i.e., beyond the base energy), including cooling, conversion, and delivery overheads. This is equivalent to consuming 150 W of dynamic power during request processing. By default, we study servers that are perfectly energy-proportional [3], i.e. they consume no base energy ($BaseCost = 0$, no need to turn machines off). Servers today are not energy-proportional, but base energy is expected to decrease considerably in the next few years (both industry and academia are seeking energy-proportionality). Nevertheless, we also study systems with different amounts of base energy ($BaseCost \neq 0$, some machines are turned off).

The default SLA we simulate requires 90% of the requests to complete in 500 ms (i.e., the processing time plus 100 ms) or less. *The SLA was satisfied at the end of the performance accounting period (one week) in all our simulations.* We study other SLAs as well.

To generate a realistic distribution of data center response times, we performed real experiments with servers located in the 3 regions mentioned above. The requests were issued from a client machine on the East Coast. Each request was made to last 400 ms on average at a remote server, according to a Poisson distribution. The client exercised the remote servers at 4 utilization levels in turn to instantiate the CDF_i tables. We leave 20% slack in utilizations, so the utilizations that delimit the ranges are really, 20%, 40%, 60%, and 80%. The time between consecutive requests issued by the client also followed a Poisson distribution with the appropriate average for the utilization level. The results of these experiments showed that higher utilizations have only a small impact on the servers' response time. Overall, we find that the servers exhibit average response times of 412 ms (East Coast), 485 ms (West Coast), and 521 ms (Europe) as measured at the client. With respect to our SLA, only the East Coast server can produce more than 90% of its replies in 500 ms or less. The other servers can only reply within 500 ms 76% (West Coast) and 16% (Europe) of the time.

The response times collected experimentally for each data center and utilization level form independent pools for our simulations. Every time a request is sent to a simulated data center, the simulator estimates the current offered load to the center and randomly selects a response time from the corresponding pool.

Electricity prices, carbon prices, and fees. We simulate two electricity prices at each data center, one for "on-peak" hours (weekdays from 8am to 8pm) and another for "off-peak" hours

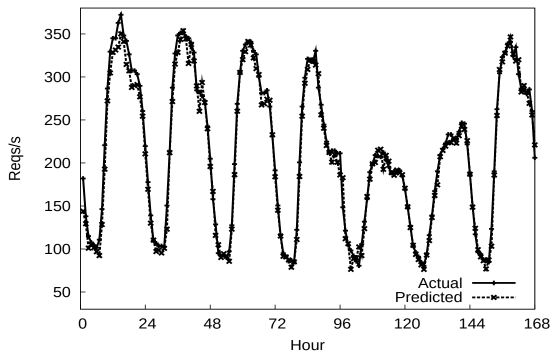

Figure 9: Actual and predicted load intensities.

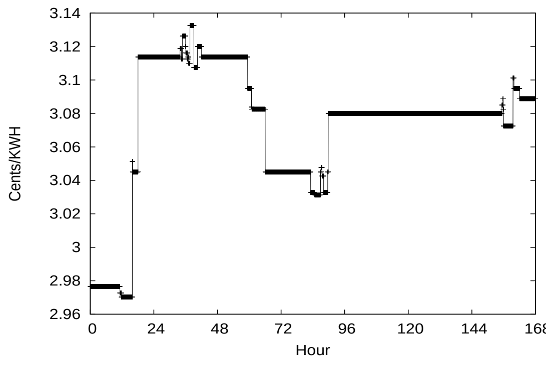

Figure 10: Carbon market prices converted to cents/KWh.

Data Center	Brown energy cost	Green energy cost
West Coast	5.23 cents/KWh	7.0 cents/KWh
East Coast	12.42 cents/KWh	18.0 cents/KWh
Europe	11.0 cents/KWh	18.0 cents/KWh

Table 3: Default electricity prices [1, 9, 20, 22, 23].

(weekdays from 8pm to 8am and weekends) [9]. The on-peak prices are listed in Table 3; by default, the off-peak prices are 1/3 of those on-peak. Note that the West Coast center is located in a state with relatively cheap electricity.

We simulate both cap-and-trade (default) and cap-and-pay. For cap-and-trade, the carbon market trace was collected from PointCarbon [25] for one month (August 2008). Figure 10 shows the carbon prices during a week, converted from Euros/ton of carbon to cents/KWh. To extend the trace to an entire year, we used the same normalization approach described above, again using data from [25]. Under cap-and-pay, the default fee for violating the brown energy cap is set at the price of brown electricity, meaning that, above the cap, the price of brown energy doubles. We also study smaller fees.

Other parameters. The default brown energy cap is equivalent to 75% of the dynamic energy required to process the trace, but we study the effect of this parameter as well. We assume that green energy can be at most 30% of the energy used at each data center. The load capacity of all data centers was assumed to be 250 requests/sec. We have scaled down the load capacity to match the intensity of our request trace.

Cost-unaware distribution. As the simplest basis for comparison, we use a cost-unaware policy (CU-Heuristic) that is similar to CA-Heuristic but disregards electricity prices and cap-violation penalties. It orders data centers according to performance, i.e. $CDF_i(L, LC_i)$, from highest to lowest. Requests are forwarded to the best-performing data center on the list until its capacity is met. At that point, new requests are forwarded to the next data center on the list and so on. Data center reconfiguration happens as in CA-Heuristic.

4.2 Real Implementation and Validation

We also implemented real prototypes of our request distribution approaches. To do so, we extended a publicly available HTTP request distribution software (called HAProxy [13]) to implement our optimization infrastructure and policies. The optimization code was taken verbatim from the simulator. The software was also modified to obey the optimized fractions in distributing the requests to the data centers. Overall, we added roughly 3K lines of new code to the roughly 18K lines of HAProxy.

Unfortunately, running the implementation in real time is impractical; computing the system's full-year results would require one full year per parameter setting. Thus, the results in Section 4.3 are simulated, but we run the real implementation for 40 hours to validate the simulator under the same assumptions.

For our validation experiments, we ran our front-end software on a machine located on the East Coast. This front-end machine was the same as our client in the response time experiments above. The data centers were also represented by the same remote servers we used in those experiments. The requests were issued to the front-end from another machine on the East Coast. This latter machine replayed a scaled-down version of two 4-hour periods during the first weekday of the Ask.com trace: a low-load period (from midnight to 4am, which includes the lowest load rate of the day) and a high-load period (from noon to 4pm, which includes the highest load rate for the day). We run each of our solution approaches and heuristics with each of the 4-hour traces. The latencies observed in these runs were fed to the simulator for a proper comparison. The experiments assumed the same default parameters as the simulator (including the same power mixes).

Our validation results are very positive. Table 4 summarizes the differences between the real executions and the simulations in terms of energy cost, brown energy consumption, and percentage of requests completed within 500 ms. The simulator and the prototype produce results that are within 2% of each other. The only exceptions are the energy costs of the two heuristics when they are exposed to a high request load. However, even in those cases, the difference is at most 6% only. The reason for the larger difference is that the simulator and the real system use different approaches for specifying the capacity of the data centers. Our simulator specifies them in requests per second, whereas the prototype uses a number of concurrent connections. Under low loads, the request traffic is not high enough for this difference in approach to noticeably alter the energy costs.

4.3 Results

4.3.1 Comparing Policies and Solution Approaches

We first compare our optimization-based policy for cap-and-trade (under different solution approaches), our cost-aware heuristic policy (CA-Heuristic), and the cost-unaware heuristic policy (CU-Heuristic). All policies and approaches use the same power mixes for the data centers. The mixes were computed by SA to

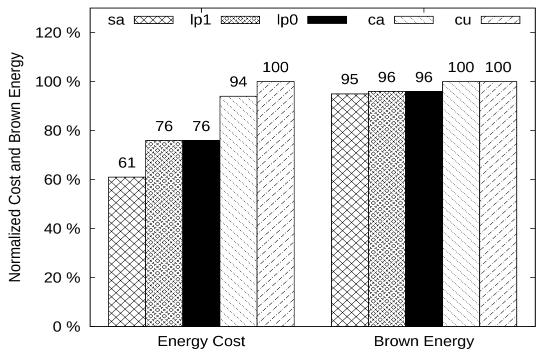

Figure 11: Comparing policies and solution approaches.

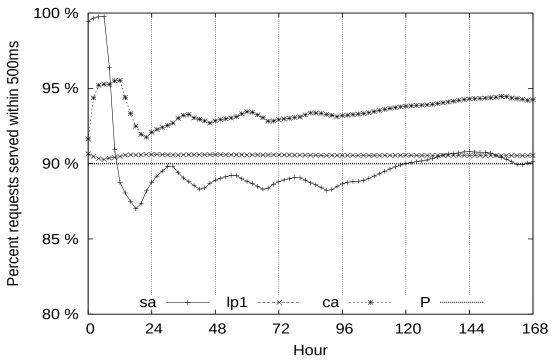

Figure 12: Detailed behavior toward meeting the SLA.

	Energy Cost	Brown Energy	SLA
Low Load			
SA	0.03%	0.07%	0.1%
LP1	0.01%	0.12%	0.39%
LP0	0.03%	0.04%	0.27%
CA	0.07%	0.09%	0.24%
CU	0.06%	0.06%	0.14%
High Load			
SA	0.81%	0.03%	0.31%
LP1	1.99%	0.42%	1.06%
LP0	2.34%	0.54%	0.96%
CA	4.14%	1.68%	0.25%
CU	5.99%	1.52%	1.7 %

Table 4: Differences between simulation and prototype.

be 80/20 (East Coast), 70/30 (West Coast), and 81/19 (Europe), where the first number in each case is the percentage of brown electricity in the mix. With these mixes, the on-peak electricity prices per KWh become 13.54 cents (East Coast), 5.76 cents (West Coast), and 12.33 cents (Europe).

Figure 11 plots the energy cost (bars on the left) and the brown energy consumption (bars on the right) for the optimization-based policy with SA, LP1, and LP0 solution approaches, CA-Heuristic (labeled "CA"), and CU-Heuristic ("CU"). The cost and consumption bars are normalized against the CU-Heuristic results.

The figure shows that both cost-aware policies produce lower costs than CU-Heuristic, as one would expect. CU-Heuristic sends the vast majority of requests to the most expensive but also best performing data center (East Coast). The figure also shows that the optimization-based policy achieves substantially (up to 35%) lower costs than CA-Heuristic, regardless of the solution approach. The reason is that CA-Heuristic tries to satisfy the SLA every hour, greedily sending requests to the most expensive data center until that happens. Comparing the solution approaches to the optimization problem, we can see that solving it for an entire week at a time as in SA leads to the lowest cost. In fact, SA achieves 39% lower costs than CU-Heuristic. Interestingly, the comparison between LP0 and LP1 shows that accurately predicting the load of the next hour is essentially useless, when solving the problem for only one hour at a time. Nevertheless, LP0 and LP1 still produce 24% lower costs than CU-Heuristic. The reason for SA's lowest cost is that it exploits the cheapest data center (West Coast) more extensively than the other approaches, as it deduces when it can compensate later for that center's lower performance (by sending requests to the best-performing data center during its off-peak times) and still meet the SLA.

To illustrate these effects, Figure 12 plots the cumulative percentage of requests serviced within 500 ms by the policies and solution approaches (LP0 almost completely overlaps with LP1, so we do not show it), as they progress towards meeting the SLA (L = 500ms, P = 90%) during the first week of the trace. SA's ability to predict future behaviors allows it to exploit low electricity prices on the West Coast (during on-peak times on the East Coast), serve slightly less than 90% of the requests within 500 ms for most of the week, and still satisfy the SLA at the end of the week. Due to their inability to predict behaviors for more than 1 hour, the other solution approaches have to be conservative about performance all the time.

Returning to Figure 11, the optimization-based policy led to only slightly lower brown energy consumptions than CA-Heuristic and CU-Heuristic. This result is not surprising, since the brown energy consumption is determined for the most part by the choice of power mix at each data center. Recall that this choice is the same for all policies and solution approaches. As we show in the next subsection, the way to conserve brown energy is to lower the brown energy cap.

Regarding the frequency of recomputations, we find that SA has to recompute a solution once every 7.6 days on average, whereas LP1 and LP0 do so roughly every 2 hours and every 1.5 hour on average, respectively. The average times to solve the optimization problem once are 392 s for SA and 1.5 ms for the LP approaches on a 3.0-GHz oct-core machine. These frequencies and times represent negligible energy overheads for the service.

These results suggest that the optimization-based policy with SA is the most cost-effective approach. The advantage of SA over LP1 and LP0 decreases as we shorten the performance accounting period. Nevertheless, SA still behaves substantially better than its linear counterparts for daily periods. Most services do not require finer granularity enforcement than a day. However, *for the few services that require tight hourly SLA guarantees, LP1 and LPO are the best choice*; although SA could be configured to produce the same results, it would do so with higher overhead.

Qureshi's heuristic. Qureshi *et al.* [28] proposed a greedy heuristic to distribute requests across data centers based on electricity prices that vary hourly. The heuristic sends requests to the cheapest data center at each point in time, up to its processing capacity, before choosing the next cheapest data center and so on. The heuristic has a very coarse control of response times by which a latency-based radius is defined around each front-end; data centers beyond the radius are not chosen as targets. We im-

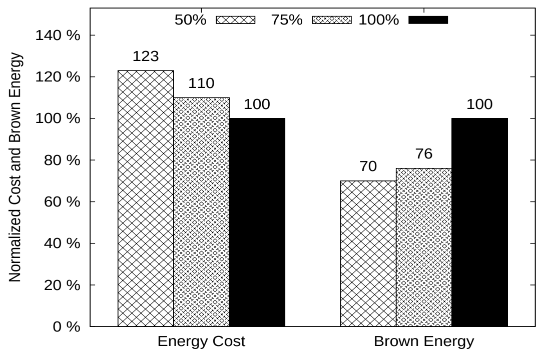

Figure 13: Effect of cap on brown energy consumption.

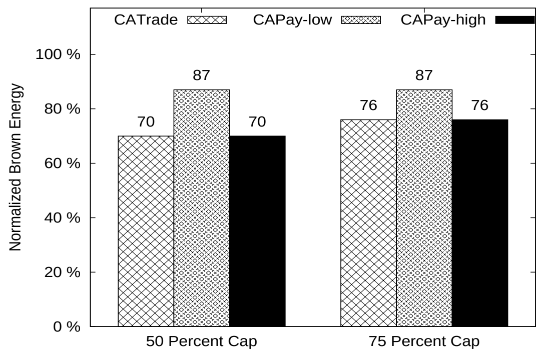

Figure 14: Comparing cap-and-trade and cap-and-pay.

plemented their heuristic for comparison and found that it either leads to very high costs or is unable to meet our SLAs, depending on the size of the radius around our East Coast front-end. When only the East Coast data center can be reached, the heuristic behaves exactly like CU-Heuristic, meeting the SLA but at a very high cost. When the West Coast data center can be reached as well, the SLA is violated because that data center is cheaper most of the time but does not meet the SLA. In this case, only 80% of the requests can be serviced within 500ms. When any data center can be reached, the situation becomes even worse, since the European data center is sometimes the cheapest but violates the SLA by a greater amount. In this latter case, the heuristic misses the SLA by almost 40%. For these reasons, we do not consider their heuristic further.

4.3.2 Conserving Brown Energy

The results we have discussed so far assume that the brown energy cap is large enough to process 75% of the requests in our trace. To understand the impact of the cap, Figure 13 displays the energy cost and brown energy consumption of SA under brown energy caps of 100% (effectively no cap), 75%, and 50% of the energy required to process the requests in the trace. The results are normalized to the no-cap scenario.

The figure shows that lowering the brown energy cap from 100% to 75% enables a savings of 24% in brown energy consumption at only a 10% increase in cost. The cost increase comes from having to pick power mixes that use more green energy; consuming brown energy beyond the cap and going to the carbon market is actually more expensive under our simulation parameters. Interestingly, decreasing the cap further to 50% increases the brown energy savings to 30% but at a much higher cost increase (24%). The reason is that there is not enough green energy to compensate for the 50% cap (the maximum amount of green energy in the power mixes is 30%). Thus, the service ends up exceeding the cap and paying the higher market costs.

These results suggest that services can significantly reduce their brown energy consumption at modest cost increases, as long as the caps are carefully picked.

4.3.3 Comparing Cap-and-Trade and Cap-and-Pay

The results we have discussed so far assume the cap-and-trade policy. Because of the relatively high cost of carbon offsets in our trace, services avoid exceeding the brown energy cap to the full extent permitted by the amount of available green energy. To understand the impact of the penalties for exceeding the cap, we now compare the cap-and-trade and cap-and-pay policies under two different fee settings for the latter policy.

Figure 14 plots the brown energy consumption under cap-and-trade (labeled “CATrade”), cap-and-pay with a low fee (“CAPay-low”), and cap-and-pay with a high fee (“CAPay-high”), as a function of the cap size. The low fee is lower than the cost of green energy, whereas the high fee is higher than that cost. All policies use SA as the solution approach.

The figure shows that CATrade and CAPay-high behave the same regardless of cap size. The reason is that they choose to consume as much green energy as possible, as discussed above. However, these policies behave quite differently than CAPay-low. Under CAPay-low, there is no incentive to use green energy, since the charge for violating the cap is actually lower than the cost of green energy. As a result, CAPay-low consumes substantially more brown energy than CATrade and CAPay-high.

These results illustrate that, if the goal is to force services to conserve brown energy, the penalties must be selected so that there is a strong incentive not to exceed the cap. Furthermore, given that carbon market prices are difficult to control, a cap-and-pay scheme may be a more effective approach to encourage brown energy conservation.

4.3.4 Sensitivity Analysis

So far, we have studied the impact of the distribution policy and solution approach, the cap size, and the capping scheme. In this subsection, we first qualitatively discuss the impact of different classes of parameters on our results. After that, we present a series of results quantifying this impact.

Qualitative discussion. After experimenting extensively with our infrastructure, we have found that four classes of parameters (besides those we evaluated above) have a strong impact on our results: (1) the relative electricity prices at the data centers over time; (2) the response time requirements imposed by the SLA; (3) the energy consumed by servers when they are idle; and (4) the percentage of writes in the workload.

The first class of parameters is important in that the electricity price differentials are the very source of possible cost savings. In addition, the availability of cheap green energy at the data centers has a direct impact on the power mixes and on our ability to conserve brown energy at low cost. Furthermore, when prices are different at each data center but constant over time, time zones are not exploited and the cost savings are limited. When on- and off-peak prices are different at each center, the ordering of data centers with respect to price changes over time. In fact, it is this

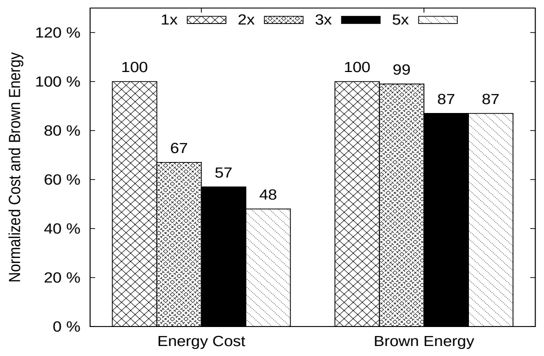

Figure 15: Effect of ratio of electricity prices.

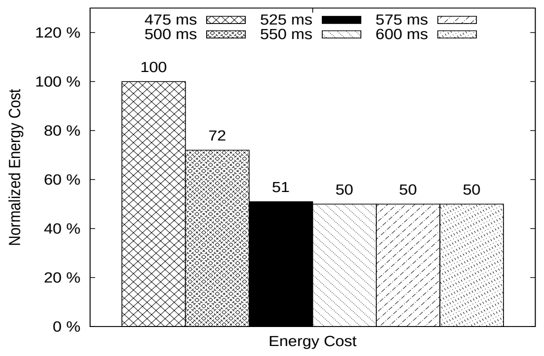

Figure 16: Effect of latency requirement (L).

ordering that really matters in defining the request distribution, not the absolute prices.

The second class is important because it determines how flexible the request distribution can be. Although this class involves many parameters (e.g., data center response times), it can be studied by varying a single knob, the SLA. Services with strict response time requirements may not be able to utilize cheap data centers that are far away from the front-end. In contrast, a front-end placed near the cheapest data center would enable all approaches (be they optimization-based or not) to easily meet the SLA at low cost; they would all use this data center most of the time. (Note however that, in practice, there would most likely be front-ends near all data centers, i.e. some front-ends would be far away from the cheapest data center as in our case study.)

The third and fourth classes determine how much of the overall energy cost can be managed through sophisticated request distribution. The third class also determines how much energy can be saved by turning servers off.

Relative electricity prices. Figure 15 plots the energy cost and brown energy consumption of the cap-and-trade policy under the SA approach, in scenarios where on-peak prices are 1x, 2x, 3x, and 5x the off-peak prices. The results are normalized against the 1x case (constant pricing). Recall that the default we have used thus far is a 3x ratio.

As one would expect, the figure shows that larger ratios reduce costs. However, the reductions are far from linear, despite the fact that there is substantially more off-peak than on-peak time during the year. The reason is that most requests (60%) are actually processed during on-peak times at the East and West Coast data centers. With respect to brown energy consumption, we see distinct behaviors at low and high ratios. At low ratios, power mixes tend to involve less green energy, since exceeding the cap is relatively inexpensive. In contrast, at high ratios, green energy becomes inexpensive a large percentage of the time.

These results suggest that services should take advantage of differentiated electricity prices to the extent that they can negotiate them with power companies.

Response time requirements. Here, we compare different response time requirements (SLAs) and the cost savings that can be achieved when they allow flexibility in request distribution. Figure 16 shows the energy costs of SA for P = 90%, as we vary L from 475 to 600ms. Recall that our results so far have assumed L = 500ms. The response times have a negligible impact on the brown energy consumption.

The figure demonstrates that having less strict SLAs enables significant cost savings, e.g. 28% and 49% for L = 500ms and 525ms, respectively. The reason is that the West Coast data center can be used more frequently in those cases. As the latency requirement is relaxed further, no more cost savings can be accrued since at that point all data centers can meet the SLA independently. We also conducted an experiment where a very strict P = 99% is enforced with L = 550ms (this is the first response time for which P = 99% can actually be satisfied). We observed that the cost compared to P = 90% increases by 16%.

These results illustrate the fundamental tradeoff between SLA requirements and our ability to lower costs. When SLAs are strict, we can meet them but at a high cost. When these requirements can be relaxed, significant cost savings can be achieved.

Base energy. Figure 17 plots the energy costs of the policies and solution approaches, as a function of the amount of power servers consume when idle. (Since we assume a dynamic power range of 150W, a base power of 150W roughly represents today's servers.) The costs are normalized to those of CU-Heuristic. Recall that all results we discussed thus far assumed no base energy.

The figure shows exactly the same trends across base powers. In fact, even under the most pessimistic assumptions, SA can still produce 20% energy cost reductions.

This result suggests that the benefits of our optimization-based framework and policies will increase with time, as servers become more energy-proportional.

Request types and on-line writes to persistent state. To assess the impact of these workload characteristics, we modified the Ask.com trace to include 3 different request types, such that one type involves writes. Our systems handle each write request at a data center by propagating coherence messages to its mirror data centers. All simulation parameters were kept at their default values, but writes were assumed to consume 60 J at each data center because of coherence activity.

Figure 18 depicts the energy costs of SA, CA-Heuristic, and CU-Heuristic, as a function of the percentage of write requests. The figure shows that SA produces costs that are 19% and 21% lower than those of CA-Heuristic and CU-Heuristic, respectively, when 20% of the requests are writes. These cost savings are smaller than those from Section 4.3.1, but are still quite significant. Write percentages that are substantially higher than 20% are not as likely in practice. Nevertheless, the SA cost savings are still 11-13% when as many as 30% of the requests are writes.

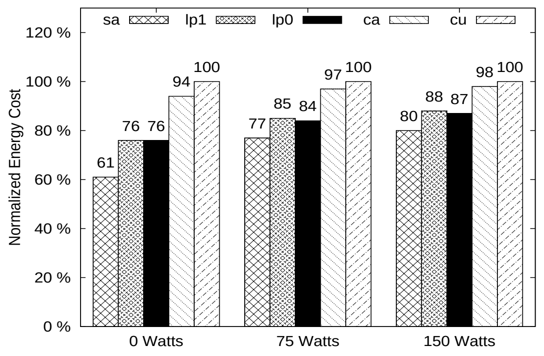

Figure 17: Effect of base energy.

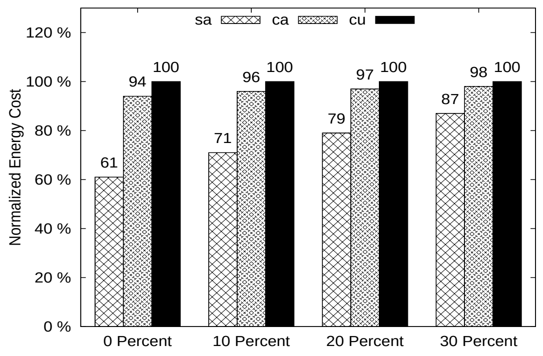

Figure 18: Effect of write percentage.

Session length. Our results so far have assumed that each session contained a single request. For completeness, we also study the impact of longer sessions. To evaluate this impact, we created two variations of the Ask.com trace that assigned each request to a specific user session. In the first variation, we created sessions with an average of 3 requests, whereas the second variation has sessions with an average of 10 requests. Recall that our systems distribute entire sessions (rather than individual requests), such that all requests belonging to a session are forwarded to the same center as the first request of the session.

Our results show that the session length has only a negligible ($\leq$ 1%) impact on energy cost and brown energy consumption. The reason is that, for any sufficiently large number of sessions, the percentage of requests of each type forwarded to a particular data center is roughly the same across all session lengths.

4.3.5 Optimizing Costs For Current Services

The results presented above considered that brown energy caps were in effect and data centers can use green energy to abide by them. However, current services do not need to cap their brown energy consumption or use green energy. This section demonstrates how our framework and policies can be used to optimize energy costs even for today's services.

Our framework enables services to take advantage of two aspects of electricity prices that they currently may not benefit from: (1) the fact that off-peak and on-peak times can have different prices; and (2) the fact that data centers at different time zones are exposed to different prices at each point in time. To isolate the benefit of these aspects, we compare our default SA results for with those of two other versions of our service. In one version, all data centers are on the East Coast with the same on- and off-peak electricity prices. To avoid changing two parameters at once, we keep the response times of the data centers the same as in the default version. This version is called EC. In the other version, the data centers are distributed like in our default configuration but each pays a fixed (but different) price for their electricity. The fixed price is the average price weighted by the hourly load intensity of the trace. This version is called FP.

Comparing our default version against EC shows that taking advantage of time zones reduces energy costs by 5%. Comparing our default version against FP shows that taking advantage of on/off prices reduces energy costs by 24%.

5 Related Work

This paper is the first to propose capping the brown energy consumption of large computer systems. It is also the first to consider carbon market interactions. Nevertheless, there have been related efforts on four topics, as we discuss next.

Conserving energy in data centers. Many works have been done on this topic, e.g. [6, 7, 8, 10, 14, 15, 21, 24, 29, 34]. Most of these works extend Load Concentration (LC) proposed in [24]. In LC, the system dynamically determines how many servers are required to serve the offered load, directs the load to this many servers, and turns the others off. Our evaluation assumed LC was performed within each data center.

Our work differs from these efforts in many ways: (1) none of the previous works considered multi-mirror services and their request distribution; (2) none of them considered the goal of capping energy consumption; and (3) none of them considered green energy or carbon market interactions.

Capping power consumption in data centers. [12, 30, 11, 38] considered power capping of data centers. Our work differs significantly from these efforts, since capping energy is qualitatively different than capping power. In particular, capping power enables cheaper cooling and packaging, and adding more hardware to a system without harming it. However, it may not conserve energy, if obeying the cap causes a significant increase in running time. In contrast, capping brown energy can conserve brown energy and promote green energy.

Optimization-based request distribution in Internet services. There have been a few previous works on this topic [19, 18, 32, 35]. Rao [32] and Shah [35] did not consider network latencies, realistic SLAs, or time-varying workloads. Moreover, these studies did not simulate or implement their proposed request distribution schemes or any competing heuristics. In contrast, [19, 18] included extensive simulations of their distribution schemes. However, none of these four works considered brown energy caps, market interactions, request types, or power mixes. In addition, none of them included a real implementation.

Importantly, note that optimization has been used in a few previous energy management works, e.g. [8, 14]. *However, none of those works considered multiple data centers or used optimization to define the load distribution.* For example, [8] optimized the number of active nodes and their CPU frequencies in a single data center. Independently of the optimization, their within-data-center distribution trivially attempts to evenly balance the load across the active nodes dedicated to a service. Our optimization computes what percentage of requests of each type should be sent to each data center.

Furthermore, our work (1) demonstrates that optimization, which is rarely used when systems must make online dynamic decisions, easily outperforms heuristics *for a completely new and timely problem, real workloads, real network latencies;* and (2) compares three solution approaches that *combine prediction and recomputation in novel ways.*

Leveraging variability in electricity prices in Internet services. Qureshi's work [27] considered variable electricity prices for each data center (the price varies on an hourly basis) and proposed to shut down entire data centers when their electricity costs are relatively high. Recently, Qureshi *et al.* [28] studied dynamic request distribution based on hourly electricity prices.

This paper makes many major contributions with respect to Qureshi's work: (1) we built a real distributed implementation for experimentation and simulator validation, whereas Qureshi relied solely on simulations; (2) we introduce brown energy caps (and carbon market interaction), whereas Qureshi did not consider caps or attempt to conserve energy; (3) we minimize costs based on caps, green energy, electricity prices, and carbon markets, whereas Qureshi focused solely on hourly electricity prices; (4) we use statistical response time information from data centers to abide by SLAs, whereas Qureshi used a simple radius metric that is unable to meet our SLAs at low cost (Section 4.3.1); and (5) we demonstrate that optimization easily outperforms heuristics such as those used by Qureshi. Contributions (2), (3), and (4) mandate significant changes in the request distribution approach and also amplify the benefits of optimization-based techniques over heuristics. *In summary, our work is critical in enabling services to manage the likely emergence of brown energy caps most effectively; a problem that Qureshi did not address at all.*

Recently, [32] extended Qureshi's work by posing and solving the same problem using optimization techniques. However, as we mentioned above, the paper did not consider energy caps, energy conservation, market interactions, network latencies, time-varying workloads, request types, or power mixes. Moreover, the evaluation of their optimized distributions was purely analytical, without simulations or a real implementation.

6 Conclusions

In this paper, we proposed an optimization-based framework for enabling multi-data-center Internet services to manage their brown energy consumption and leverage green energy, while respecting their SLAs and minimizing costs. Moreover, our framework enables services to exploit different electricity prices and data centers located at different time zones to minimize costs, even in the absence of brown energy caps, different electricity prices, or green energy. We also proposed a simple heuristic for achieving the same goals. We evaluated our proposals extensively, using simulations, real experiments, and real traces.

Our work demonstrates the value of optimization techniques, workload prediction, and electricity price diversity for energy management in Internet services. Given current high energy costs, as well as likely future caps on carbon emissions and/or brown energy consumption, frameworks such as ours should become extremely useful in practice.

Acknowledgements

We would like to thank Fred Chong and Luiz Barroso for comments that helped improve this paper. We would also like to thank Frank Felder and Robert Patrick for helping us understand the carbon and electricity markets. Last but not least, we would like to thank Ask.com for giving us a real request trace, and Dejan Kostic and John Zahorjan for lending us machines at EPFL and the University of Washington, respectively.

References

[1] Energy Information Administration. Average Retail Price of Electricity to Ultimate Customers by End-Use Sector, by State. http://www.eia.doe.gov/cneaf/electricity/epm/table5_6_b.html.

[2] US Environmental Protection Agency. EPA Report on Server and Data Center Energy Efficiency. August 2007.

[3] L. A. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 40(12), December 2007.

[4] G. Box and G. Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990.

[5] V. Cardellini, M. Colajanni, and P. Yu. Dynamic Load Balancing on Web-Server Systems. *IEEE Internet Computing*, 3(3), May 1999.

[6] J. Chase et al. Managing Energy and Server Resources in Hosting Centers. In *SOSP*, October 2001.

[7] G. Chen et al. Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services. In *NSDI*, April 2008.

[8] Y. Chen et al. Managing Server Energy and Operational Costs in Hosting Centers. In *SIGMETRICS*, June 2005.

[9] K. Coughlin et al. The Tariff Analysis Project: A Database and Analysis Platform for Electricity Tariffs. http://repositories.cdlib.org/lbnl/LBNL-55680.

[10] E. N. Elnozahy, M. Kistler, and R. Rajamony. Energy Conservation Policies for Web Servers. In *USITS*, March 2003.

[11] X. Fan, W.-D. Weber, and L. A. Barroso. Power Provisioning for a Warehouse-sized Computer. In *ISCA*, June 2007.

[12] M. Femal and V. Freeh. Boosting Data Center Performance Through Non-Uniform Power Allocation. In *ICAC*, June 2005.

[13] HAProxy. The Reliable, High Performance TCP/HTTP Load Balancer. http://haproxy.1wt.eu/.

[14] T. Heath et al. Energy Conservation in Heterogeneous Server Clusters. In *PPoPP*, June 2005.

[15] T. Horvath et al. Dynamic Voltage Scaling in Multi-tier Web Servers with End-to-End Delay Control. *IEEE Transactions on Computers*, 56(4), April 2007.

[16] Uptime Institute and McKinsey & Company. Revolutionizing Data Center Energy Efficiency Key Analyses. 2008.

[17] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598), May 1983.

[18] K. Le et al. A Cost-Effective Distributed File Service with QoS Guarantees. In *Middleware*, November 2007.

[19] K. Le et al. Cost- And Energy-Aware Load Distribution Across Data Centers. In *HotPower*, October 2009.

[20] Green Electricity Marketplace. Green Electricity Marketplace. http://www.greenelectricity.org/.

[21] D. Meisner, B. T. Gold, and T. F. Wenisch. PowerNap: Eliminating Server Idle Power. In *ASPLOS*, March 2009.

[22] The Green Power Network. Buy Green Power. http://apps3.eere.energy.gov/greenpower/buying/buying_power.shtml.

[23] Department of Energy and Climate Change. Quarterly Energy Prices. http://stats.berr.gov.uk/uksa/energy/sa20090625b.htm.

[24] E. Pinheiro et al. Dynamic Cluster Reconfiguration for Power and Performance. In L. Benini, M. Kandemir, and J. Ramanujam, editors, *Compilers and Operating Systems for Low Power*. Kluwer Academic Publishers, August 2003. Earlier version published in COLP, September 2001.

[25] PointCarbon, Retrieved in March 2008. http://www.pointcarbon.com.

[26] Power Scorecard. Electricity from Coal, Retrieved in April 2008. http://www.powerscorecard.org/tech_detail.cfm?resource_id=2.

[27] A. Qureshi. Plugging Into Energy Market Diversity. In *HotNets*, October 2008.

[28] A. Qureshi et al. Cutting the Electric Bill for Internet-Scale Systems. In *SIGCOMM*, August 2009.

[29] K. Rajamani and C. Lefurgy. On Evaluating Request-Distribution Schemes for Saving Energy in Server Clusters. In *ISPASS*, March 2003.

[30] P. Ranganathan et al. Ensemble-Level Power Management for Dense Blade Servers. In *ISCA*, June 2006.

[31] S. Ranjan, R. Karrer, and E. Knightly. Wide Area Redirection of Dynamic Content by Internet Data Centers. In *INFOCOM*, March 2004.

[32] L. Rao et al. Minimizing Electricity Cost: Optimization of Distributed Internet Data Centers in a Multi-Electricity-Market Environment. In *INFOCOM*, March 2010.

[33] Regional Greenhouse Gas Initiative. Regional Greenhouse Gas Initiative: An Initiative of the Northeast and Mid-Atlantic States of the US, Retrieved in April 2008. http://www.rggi.org/.

[34] C. Rusu et al. Energy-Efficient Real-Time Heterogeneous Server Clusters. In *RTAS*, April 2006.

[35] A. J. Shah and N. Krishnan. Optimization of Global Data Center Thermal Management Workload for Minimal Environmental and Economic Burden. *IEEE Transactions on Components and Packaging Technologies*, 31(1), March 2008.

[36] UK Government. Carbon Reduction Commitment, Retrieved in July 2009. http://www.carbonreductioncommitment.info/.

[37] L. Wang, V. Pai, and L. Peterson. The Effectiveness of Request Redirection on CDN Robustness. In *OSDI*, December 2002.

[38] X. Wang and M. Chen. Cluster-level Feedback Power Control for Performance Optimization. In *HPCA*, February 2008.

[39] W. Zhao, D. Olshefski, and H. Schulzrinne. Internet Quality of Service: An Overview. Technical Report CUCS-003-00, Department of Computer Science, Columbia University, February 2000.

Appendix A: Extended framework and cap-and-trade formulation including sessions and writes

Symbol	Meaning
$f_i^y(t)$	Percentage of requests of type y to be forwarded to center i in epoch t
m_i^{brown}	Percentage of brown electricity in the power mix of center i
m_i^{green}	Percentage of green electricity in the power mix of center i
Overall Cost	Total energy cost (in $) of the service
BEC	Brown energy cap (in KWh) of all centers combined
$p^y(t)$	Percentage of requests of type y in the workload in epoch t
$w^y(t)$	Percentage of requests of type y in epoch t that involve writes to persistent state
M_i^y	Equals 1 if data center i can serve requests of type y. Equals 0 otherwise
$ReqCost_i^y(t)$	Avg. energy cost (in $) incl. cap-violation charges of serving a request of type y at center i in epoch t
$CohCost_i^y(t)$	Avg. energy cost (in $) incl. cap-violation charges of serving a coherence request of type y at center i in epoch t
$BaseCost_i(\mathit{offered}_i, t)$	Base energy cost (in $) incl. cap-violation charges of center i in epoch t, under $\mathit{offered}_i$ load
$br_i^y(t)$	Avg. energy cost (in $) of serving a request of type y using only brown electricity at center i
$gr_i^y(t)$	Avg. energy cost (in $) of serving a request of type y using only green electricity at center i
$bc_i^y(t)$	Avg. energy cost (in $) of serving a coherence request of type y using only brown electricity at center i
$gc_i^y(t)$	Avg. energy cost (in $) of serving a coherence request of type y using only green electricity at center i
$bb_i(\mathit{offered}_i, t)$	Base energy cost (in $) of center i in epoch t, under $\mathit{offered}_i$ load, using only brown electricity
$gb_i(\mathit{offered}_i, t)$	Base energy cost (in $) of center i in epoch t, under $\mathit{offered}_i$ load, using only green electricity
$\mathit{marketReq}_i^y(t)$	Avg. market cost (in $) to offset the brown energy used by a request of type y in center i in epoch t
$\mathit{marketCoh}_i^y(t)$	Avg. market cost (in $) to offset the brown energy used by a coherence request of type y in center i in epoch t
$\mathit{marketBase}_i(\mathit{offered}_i, t)$	Market cost (in $) of offsetting the base brown energy in center i in epoch t, under $\mathit{offered}_i$ load
$\mathit{feeReq}_i^y(t)$	Avg. fee for cap violation (in $) for the brown energy used by a request of type y in center i in epoch t
$\mathit{feeCoh}_i^y(t)$	Avg. fee for cap violation (in $) for the brown energy used by a coherence request of type y in center i in epoch t
$\mathit{feeBase}_i(\mathit{offered}_i, t)$	Fee for cap violation (in $) for the base brown energy in center i in epoch t, under $\mathit{offered}_i$ load
LR(t)	Expected peak service load rate (in reqs/sec) for epoch t
LT(t)	Expected total service load (in number of reqs) for epoch t
LC_i	Load capacity (in reqs/sec) of center i
$CDF_i(L, \mathit{offered}_i)$	Expected percentage of requests sent to center i that complete within L time, given $\mathit{offered}_i$ load

Table 5: Extended framework. Note that this table extends Table 1 with the parameters associated with write requests and the coherence activity that they trigger: $w^y(t)$, $CohCost_i^y(t)$, $bc_i^y(t)$, $gc_i^y(t)$, $\mathit{marketCoh}_i^y(t)$, and $\mathit{feeCoh}_i^y(t)$.

$$\begin{aligned}\mathit{offered}_i = \;& (\textstyle\sum_y f_i^y(t) \times M_i^y \times p^y(t) \times LR(t)) + (\sum_y (1 - f_i^y(t)) \times w^y(t) \times M_i^y \times p^y(t) \times LR(t)) = \\ & \textstyle\sum_y (f_i^y(t) + ((1 - f_i^y(t)) \times w^y(t))) \times M_i^y \times p^y(t) \times LR(t)\end{aligned} \tag{5}$$

$$\begin{aligned}\mathit{Overall\ Cost} = \;& (\textstyle\sum_t \sum_i \sum_y f_i^y(t) \times M_i^y \times p^y(t) \times LT(t) \times \mathit{ReqCost}_i^y(t)) + \\ & (\textstyle\sum_t \sum_i \sum_y (1 - f_i^y(t)) \times w^y(t) \times M_i^y \times p^y(t) \times LT(t) \times \mathit{CohCost}_i^y(t)) + \\ & (\textstyle\sum_t \sum_i BaseCost_i(\mathit{offered}_i, t))\end{aligned} \tag{6}$$

$$\begin{aligned}ReqCost_i^y(t) = \;& m_i^{brown} \times br_i^y(t) + m_i^{green} \times gr_i^y(t), \textit{if brown energy consumed so far } \leq BEC \\ & marketReq_i^y(t) + m_i^{brown} \times br_i^y(t) + m_i^{green} \times gr_i^y(t), \textit{otherwise}\end{aligned} \tag{7}$$

$$\begin{aligned}CohCost_i^y(t) = \;& m_i^{brown} \times bc_i^y(t) + m_i^{green} \times gc_i^y(t), \textit{if brown energy consumed so far } \leq BEC \\ & marketCoh_i^y(t) + m_i^{brown} \times bc_i^y(t) + m_i^{green} \times gc_i^y(t), \textit{otherwise}\end{aligned} \tag{8}$$

$$\begin{aligned}BaseCost_i(\mathit{offered}_i, t) = \;& m_i^{brown} \times bb_i(\mathit{offered}_i, t) + m_i^{green} \times gb_i(\mathit{offered}_i, t), \textit{if brown energy consumed so far } \leq BEC \\ & marketBase_i(\mathit{offered}_i, t) + m_i^{brown} \times bb_i(\mathit{offered}_i, t) + m_i^{green} \times gb_i(\mathit{offered}_i, t), \textit{otherwise}\end{aligned} \tag{9}$$

$$\begin{aligned}&\text{1. } \forall i\; m_i^{brown} \textit{ and } m_i^{green} \geq 0 \Rightarrow \textit{i.e., each part of the mix cannot be negative.}\\ &\text{2. } \forall i\; m_i^{brown} + m_i^{green} = 1 \Rightarrow \textit{i.e., the mixes need to add up to 1.}\\ &\text{3. } \forall t \forall i \forall y\; f_i^y(t) \geq 0 \Rightarrow \textit{i.e., each fraction cannot be negative.}\\ &\text{4. } \forall t \forall y\; \textstyle\sum_i f_i^y(t) \times M_i^y = 1 \Rightarrow \textit{i.e., the fractions for each request type need to add up to 1.}\\ &\text{5. } \forall t \forall i\; \mathit{offered}_i \leq LC_i \Rightarrow \textit{i.e., the offered load to a data center should not overload it.}\\ &\text{6. } \textstyle\sum_t \sum_i \sum_y (f_i^y(t) \times M_i^y \times p^y(t) \times LT(t) \times CDF_i(L, \mathit{offered}_i)) \,/\, \sum_t LT(t) \geq P \Rightarrow \textit{i.e., the SLA must be satisfied.}\end{aligned} \tag{10}$$

Figure 19: Extended cap-and-trade formulation. These equations extend those from Figure 5 by (1) accounting for data coherence activity when defining *Overall Cost*; and (2) defining the average energy cost of the coherence activity at a mirror data center ($CohCost_i^y(t)$), as a result of a write processed at another data center. The equations also re-define the offered load to a data center ($\mathit{offered}_i$). The constraints are unchanged from their definition in Figure 5.