Cost- and Energy-Aware Load Distribution Across Data Centers

Kien Le[†], Ricardo Bianchini[†], Margaret Martonosi[‡], and Thu D. Nguyen[†]
[†]Rutgers University [‡]Princeton University

1 Introduction

Today, many large organizations operate multiple data centers. The reasons for this include natural business distribution, the need for high availability and disaster tolerance, the sheer size of their computational infrastructure, and/or the desire to provide uniform access times to the infrastructure from widely distributed client sites. Regardless of the reason, these organizations consume significant amounts of energy and this energy consumption has both a financial and environmental cost.

Interestingly, the geographical distribution of the data centers often exposes many opportunities for optimizing energy consumption and costs by intelligently distributing the computational workload. We are interested in three such opportunities. First, we seek to exploit data centers that pay different and perhaps variable electricity prices. In fact, many power utilities now allow consumers to choose hourly pricing, e.g. [1]. Second, we seek to exploit data centers that are located in different time zones, which adds an extra component to price variability. For example, one data center may be under peak-demand prices while others are under off-peak-demand prices. Third, we seek to exploit data centers located near sites that produce renewable (hereafter called "green") electricity to reduce "brown" energy consumption that is mostly produced by carbon-intensive means, such as coal-fired power plants.

To make our investigation of these degrees of freedom more concrete, in this paper we consider multi-data-center Internet services, such as Google or iTunes. These services place their data centers behind a set of front-end devices. The front-ends are responsible for inspecting each client request and forwarding 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 paper proposes and evaluates a framework for optimization-based request distribution. The framework enables services to manage their energy consumption and costs, while respecting their SLAs. It also allows services to take full advantage of the degrees of freedom mentioned above. Based on the framework, we propose two request distribution policies. For comparison, we also propose a greedy heuristic designed with the same goals and constraints as the other policies.

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. After fractions are computed, the front-ends abide by them until they are recomputed. The heuristic policy 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.

Our evaluation uses a day-long trace from a commercial service. Our results show that the optimization-based policies can accrue substantial cost reductions by intelligently leveraging time zones and hourly electricity prices. The results also show that we can exploit green energy to achieve significant reductions in brown energy consumption for small increases in cost.

Related work. The vast majority of the previous work on data center energy management has focused on a single data center. We are not aware of any previous work that addresses load distribution across data centers with respect to their energy consumption or energy costs. Moreover, we are not aware of other works on leveraging time zones, variable electricity prices, or green energy sources. The exception here is [10], which leverages electricity price diversity to shut down entire data centers when their electricity costs are relatively high. Finally, we know of no previous work on optimization-based request distribution in Internet services, besides our own [8]. However, our previous work did not address energy issues, time zones, or heuristics at all.

2 Request Distribution Policies

We assume that a front-end is chosen to first handle a client request via round-robin DNS or some other high-level policy. The front-ends execute one of our policies and forward each request to a data center that can serve it. Typically, a request can only be served by 2 or 3 *mirror* data centers; further replicating content would increase the state-consistency traffic without a meaningful benefit

$$\textit{Overall Cost} = (\sum_t \sum_i f_i(t) \times LT(t) \times \textit{Cost}_i(t)) + (\sum_t \sum_i \textit{BCost}_i(\textit{offered}_i, t)) \tag{1}$$

$$\begin{array}{ll} \textit{Policy EPrice:} & Cost_i(t) = c_i(t) \textit{ and } BCost_i(\textit{offered}_i, t) = b_i(\textit{offered}_i, t) \\ \textit{Policy GreenDC:} & Cost_i(t) = c_i^{green}(t) \textit{ and } BCost_i(\textit{offered}_i, t) = b_i^{green}(\textit{offered}_i, t), \textit{ if green energy consumed so far } \leq GE_i \\ & Cost_i(t) = c_i(t) \textit{ and } BCost_i(\textit{offered}_i, t) = b_i(\textit{offered}_i, t), \textit{ otherwise} \end{array} \tag{2}$$

$$\begin{array}{l} \textit{1. } \forall t \forall i \; f_i(t) \geq 0 \Rightarrow \textit{i.e., each fraction cannot be negative.} \\ \textit{2. } \forall t \sum_i f_i(t) = 1 \Rightarrow \textit{i.e., the fractions for each request type need to add up to 1.} \\ \textit{3. } \forall t \forall i \; (f_i(t) \times LR(t)) \leq LC_i \Rightarrow \textit{i.e., the offered load to a data center should not overload it.} \\ \textit{4. } \sum_t \sum_i (f_i(t) \times LT(t) \times CDF_i(L, \textit{offered}_i)) \,/\, \sum_t LT(t) \geq P \Rightarrow \textit{i.e., the SLA must be satisfied.} \end{array} \tag{3}$$

Symbol	Meaning
$f_i(t)$	% requests to be forwarded to center i
Overall Cost	Total energy cost ($)
$Cost_i(t)$, $c_i(t)$	Avg. cost ($) of a request at center i
$c_i^{green}(t)$	Avg. cost ($) of a request at center i using green energy
$BCost_i(\textit{offered}_i, t)$, $b_i(\textit{offered}_i, t)$, $b_i^{green}(\textit{offered}_i, t)$	Base energy costs ($) of center i under *offered*$_i$ load
GE_i	Amount of green energy that green center i can consume
LC_i	Load capacity (reqs/sec) of center i
LR(t)	Expected peak service rate (reqs/sec)
LT(t)	Expected total service load (#reqs)
offered$_i$	LR(t) times $f_i(t)$ (reqs/sec)
$CDF_i(L, \textit{offered}_i)$	Expected % requests that complete within L time, given *offered*$_i$ load

Table 1: Framework parameters. (t) represents time.

in availability or performance. The reply is sent to the original front-end, which in turn forwards it to the client.

2.1 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. Our policies respect these requirements by having the front-ends: (1) prevent data center overloads; and (2) monitor the response time of the data centers, and adjust the request distribution to correct any performance or availability problems.

We assume that the service has a single SLA with its customers, which is enforced on a daily basis, the "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-end devices*. The SLA guarantee provided by our policies and framework can be combined with Internet QoS approaches to achieve end-to-end guarantees [11].

Note the SLA definition 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 accounting period.

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 for unexpected increases in load); other servers can be turned off, as in [4, 5, 6, 9].

2.2 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 find the fraction $f_i(t)$ of requests that should be sent to each mirror data center i, during "epoch" t. (An epoch is defined as a period of fixed fractions. There can be many epochs during a single accounting period.) The next subsection describes two specific optimization problems (policies). Section 2.2.2 describes the instantiation of the parameters. Section 2.2.3 discusses how to solve the problems.

2.2.1 Problem Formulations

Policy EPrice: Leveraging time zones and variable electricity prices. The $f_i(t)$ fractions should minimize the overall energy cost, *Overall Cost*. Equation 1 defines *Overall Cost* with two additive components. The first represents the energy cost of processing the client requests that are offered to the service. The second represents the "base" energy cost, i.e. the cost of the energy that is spent when the active servers are idle. In the EPrice policy, the per-request $Cost_i$ and the base energy cost $BCost_i$ have trivial definitions (Equation 2). *Overall Cost* should be minimized under the constraints that follow the equations.

Policy GreenDC: Leveraging data centers powered by green energy. The formulation above does not distinguish data centers based on their energy source. However, we expect that data centers will increasingly often be located near sources of green energy, such as wind and solar farms. In this scenario, the same service could have some data centers that are powered by brown energy (brown data centers), and others that are powered by green energy (green data centers). Because the supply of green energy may not be enough to power a data center throughout the entire period, green data centers must also be connected to the regular electrical grid.

To formalize this scenario, we can redefine $Cost_i$ and $BCost_i$ for the green data centers as in the GreenDC policy (Equation 2), where GE_i is the amount of green energy that green center i can consume during the accounting period. The definitions of $Cost_i$ and $BCost_i$ for brown data centers stay the same as before.

Other options. We have not yet explored services with session state, i.e. soft state that only lasts a user's session with the service. In such services, the distribution is constrained since all requests of a session must be sent to the same data center. Nevertheless, it is easy to extend our work to handle sessions by (1) estimating the average number of requests per session; and (2) computing fractions that guide the distribution of the first request of a session. It is also fairly easy to handle (1) requests that involve writes to persistent state and (2) multiple request types, instead of averaging across all types like we do now. We will explore these issues in our future work.

2.2.2 Instantiating Parameters

To instantiate the parameters of our formulations 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 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> entries for each data center that is filled over time. Similarly, we create a table of <offered load, base energy cost> entries to instantiate $BCost_i$.

2.2.3 Solving the Optimization Problem

The solution for an entire accounting period provides the best energy cost. However, such a solution is only possible when we can predict future load intensities, electricity prices, and data center response time distributions (CDF_i). Electricity price predictions are trivial when the price is constant or when there are two prices. When prices vary hourly, we can use the day-ahead prediction that is provided by the utility for each day [1]. Typically, these day-ahead prices are reasonably good predictions of actual prices. For predicting load intensities, we consider Auto-Regressive Moving Average (ARMA) modeling [3]. We do not attempt to predict CDF_i. Instead, we assume the current CDF_i tables as predictions.

Characteristic	Optimization (EPrice & GreenDC)	CA-Heuristic
Accounting period	1 day	1 day
Epoch length	4 hours	1 hour
Load predictions	Per front-end for entire day	Per front-end for next hour
Energy price predictions	Entire day	Next hour
Recomputation decision	Epoch boundary	Epoch boundary
Communication with DCs	Yes	Yes

Table 2: Main characteristics of distribution approaches.

We cannot use fast Linear Programming (LP) solvers, because solving for an entire day at once involves a few non-linear functions (e.g., $BCost_i$ and CDF_i). Instead of LP, we use Simulated Annealing [7] and divide the day into six 4-hour epochs, i.e. $t = 1..6$. We will consider running an LP solver every hour in our future work.

Because the assumptions/predictions that we make/use when computing a solution may become invalid/inaccurate over time, we must check for deviations. If there is any significant deviation at an epoch boundary, we recompute the solution. We must also recompute if a data center becomes unavailable (or, in our second formulation, the green energy expires at a data center). In practice, recomputations occur at the granularity of multiple hours.

After a recomputation and every hour, the front-ends inform the data centers about their predicted loads for the next hour, so that they can reconfigure. The "Optimization" column of Table 2 summarizes our approach.

2.3 Heuristics-Based Request Distribution

We also propose a heuristic policy (CA-Heuristic) that is still cost-aware, but is simpler and less computationally intensive than the optimization-based approach. The heuristic is greedy and uses 1-hour epochs. 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 ARMA for each front-end. 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. The remaining data centers are ordered by the same ratio. A final list, called *MainOrder*, is created by concatenating the two lists of data centers.

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

Data Center	Brown energy (cents/KWh)	Green energy (cents/KWh)	Capacity (reqs/s)
DC 1 (West US)	11.1	15.0 (solar)	125
DC 2 (East US)	11.7	—	215
DC 3 (Europe)	9.7	8.0 (wind)	125

Table 3: Default simulation parameters. Capacities have been scaled down to match our request trace.

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 about their predicted loads for the next epoch. The last column of Table 2 overviews our heuristic.

3 Evaluation

3.1 Methodology

We implemented a simulator of a large Internet service. For simplicity, we simulate a single front-end 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 Central Europe.

Request trace. Our day-long trace is a representative fraction of the requests received by Ask.com. Figure 1 shows the 90th percentile of the actual and ARMA-predicted request rates during each hour. The figure shows that the ARMA predictions are very accurate.

Our trace does not include response times. To generate realistic data center response times, we installed a simple service on 3 PlanetLab machines in the right time zones. The requests were made from a machine at Rutgers, i.e. our front-end. We assume that the average raw processing time of each request is 200 ms. To mimic the effect of load intensity and network congestion, we increase the response times based on the load offered to each center (5% increase in time for each 25% increase in load).

Electricity prices, sources, and time zones. We simulate schemes with one electricity price, two prices (on/off peak), and hourly prices. For the on/off-peak scheme (On/Off), off-peak hours are from 9pm to 7am. For the hourly scheme (Dynamic), Figure 2 shows the day-ahead and actual brown electricity prices we use [1]. The day-ahead prices predict trends fairly accurately, but not absolute prices. To mimic different brown electricity prices for each data center, we simply shift our default prices 3 hours earlier or 6 hours later. To make all pricing schemes comparable, the prices for the constant and on/off-peak schemes are computed based on the real prices in Figure 2. When we consider green data centers, their electricity price is always assumed constant. We also assume that the amount of green energy available daily at each green site is enough to process 25% of the requests in the trace. The constant brown and green prices are listed in Table 3.

Other parameters. We assume that a request consumes 60 J of dynamic energy to process by 2 machines, including cooling, conversion, and delivery overheads. This is equivalent to consuming 150 W of dynamic power per machine during request processing. By default, we study machines that are fully energy-proportional [2], i.e. they consume no base energy. In addition, we study the impact of this assumption. The SLA requires 90% of the requests to complete in 700 ms (processing time plus 500 ms) or less. *The SLA was satisfied at the end of the accounting period (one day) in all our simulations.* Table 3 lists the data center capacities.

Cost-unaware distribution. As the simplest basis for comparison, we use a cost-unaware policy (CU-Heuristic) that is similar to CA-Heuristic. It orders data centers according to performance, i.e. $CDF_i(L, LC_i)$, from highest to lowest. Requests are forwarded to the first 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.

3.2 Results

Effect of cost-awareness and pricing scheme. Figure 3 depicts the energy cost of the EPrice, CA-Heuristic, and CU-Heuristic policies under the three (brown) electricity pricing schemes. The figure shows many important results: (1) As expected, both cost-aware policies reduce costs compared to CU-Heuristic, even under constant pricing; (2) the On/Off and Dynamic schemes enable significant cost reductions compared to constant pricing, especially under EPrice; and (3) EPrice always achieves lower cost than CA-Heuristic. In fact, combining cost-awareness and dynamic pricing enables EPrice to reduce cost by 25%. In general, EPrice behaves better than CA-Heuristic because it often uses the cheapest but worst-performing data center (Europe), instead of the expensive but best-performing data center (US East Coast). The reason is that EPrice predicts that it can compensate for the poor performance of the European data center during future periods of low load.

Effect of time zones. Figure 3 assumes that each data center is in a different time zone. When this is not the case, serving a request costs the same at any data center. For this reason, all policies achieve the same cost, regardless of pricing scheme. This cost is slightly higher than that of CU-Heuristic in Figure 3, suggesting that multiple time zones are critical to enable cost savings.

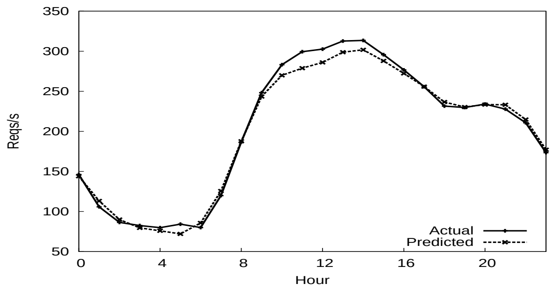

Figure 1: Actual and predicted load intensities.

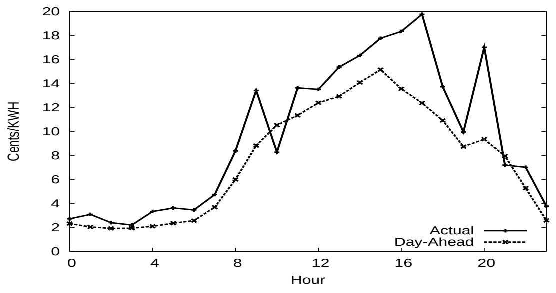

Figure 2: Actual and day-ahead brown electricity prices.

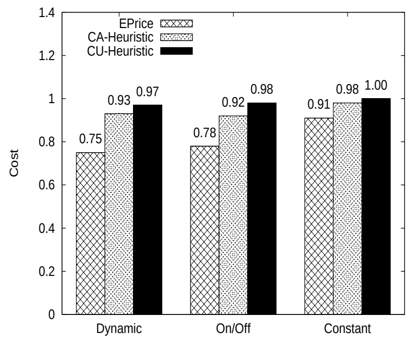

Figure 3: Pricing and cost-awareness.

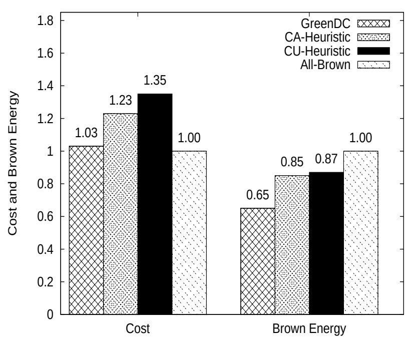

Figure 4: Green data centers.

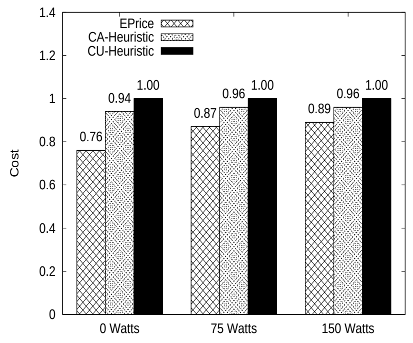

Figure 5: Base energy.

Effect of green data centers. Figure 4 depicts the cost and brown energy consumption of the GreenDC policy, CA-Heuristic, and CU-Heuristic, under dynamic pricing. The results are normalized against the default results for EPrice with dynamic pricing ("All-Brown"), i.e. the leftmost bar in Figure 3. Figure 4 shows that GreenDC can decrease brown energy consumption by 35% by leveraging the green data centers at only a 3% cost increase. The heuristic policies save substantially less brown energy at much higher costs than GreenDC. Again, the reason is that the heuristic policies often use the East Coast data center, instead of the wind-based European data center.

Effect of base energy. The results above all assume that servers do not consume any power when idle. Figure 5 quantifies the effect of the base energy by comparing the default results for EPrice, CA-Heuristic, and CU-Heuristic to those when a server consumes 75W and 150W when idle. We assume that no data center consumes green energy. This figure shows that increasing the base energy reduces the cost savings achievable by our optimization approach. The gains are smallest (but still non-trivial) for Base = 150W. This result shows that the benefits of our approach will increase with time, as servers become more energy-proportional.

4 Conclusions

In this paper, we proposed a framework for optimization-based request distribution in multi-data-center Internet services. We also proposed two policies for managing these services' energy consumption and cost, while respecting their SLAs. The policies take advantage of time zones, variable electricity prices, and green energy. Finally, we proposed a simple heuristic for achieving the same goals. Our evaluation showed positive results.

References

[1] Ameren. Real-time Prices. https://www2.ameren.com/-RetailEnergy/realtimeprices.aspx.

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

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

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

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

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

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

[8] K. Le, R. Bianchini, and T. D. Nguyen. A Cost-Effective Distributed File Service with QoS Guarantees. In *Proceedings of Middleware*, November 2007.

[9] E. Pinheiro et al. Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems. In *Proceedings of COLP*, September 2001.

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

[11] 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.