

Experimental Evaluation of an Energy-Delay Aware Web Routing Method

Gandhimathi Velusamy
Department of Computer Science
University of Houston
Houston, Texas-77204
Email: gvelusamy@uh.edu

Ricardo Lent
College of Technology
University of Houston
Houston, Texas-77204
Email: rlent@uh.edu

Abstract—The web infrastructure continues to grow in both size and diversity and is quickly approaching 2 billion sites that serve an equally massive number of users. Content replication is a common technique that aims to improve reliability and performance. It consists of the use of multiple server mirrors that operate across geographically distributed regions. Several factors impair the static system optimization or even a periodic optimization. Final energy costs depend on the level of workload handled by each region given local variations in energy pricing. To accentuate the problem, energy pricing may dynamically change for some sites. Also, network state fluctuations, that are produced by congestion and failures, create time-varying performance to different users. These factors contribute to affect user experience and service costs. We propose an energy-delay aware web routing method that dynamically directs user requests to a set of replicated sites. The method relies on learning to implement a customized performance-cost load-balancing. We present experimental results from a network testbed using actual power measurements and simulated spatial variations in the energy prices according to the U.S. electricity market. The results show that the method can help to achieve the desired balance of energy cost and average response time, while reducing energy costs up to 11% without introducing a major impact to service quality.

I. INTRODUCTION

Today's landscape for information technologies in the enterprise includes a mixed use of edge and multiple clouds that involve the use of private and public infrastructure (Data Centers). Large enterprises seek their business to be spread across the globe: offering accessible services, supporting flex work times for employees, etc. Typical cases include the use of Enterprise Resource Planning Software such as Systems Applications and Products (SAP), Customer Relationship Management (CRM), PeopleSoft, Business Intelligence (BI) etc., which help to streamline business processes. In the past, these applications were deployed on premises, but there is an increasing push for cloud adoption and the use of web services. Regardless of the deployment method, client requests are expected to receive optimal delay as many of these applications are interactive and hence, sensitive to delay. Client traffic is unpredictable and dynamic because of the random behavior of users requesting services such as flash crowds.

As more and more technical innovations and sophistication are introduced into businesses, the use of computing resources tend to increase manifold and eventually consuming enormous amount of electricity. It has been estimated that the energy consumed by Data Centers in the United States in 2020 will be 140 billion kWh, which is equivalent to the power generated by 50 power plants and will cause 150 tons of carbon emissions [31]. Beyond the ecological concerns, the increase in energy usage directly impacts operating costs. Electricity prices depend on multiple factors and are not uniform across the globe.

In the USA, the price depends on the availability of power plants, fuel costs, and pricing regulations by the 10 Regional Transmission Organizations (RTO). Usually the client demand for electricity also contributes to price variations. So prices may be higher during peak usage hours. By avoiding 1% peak demand, 10% of generated electricity could be saved [8]. Although not regulated, the Demand-Response (DR) program makes prices variable to encourage a reduction in the electricity usage during peak hours [2]. In this program, electricity charges are decided according to changes in the electricity market over the time. DR schemes are classified as: 1) Incentive-based programs in which, customers are rewarded with rebates or incentives during peak usage if they reduce their consumption. 2) Price-based programs, which are based on dynamic pricing rates such as *Time-of-Usage(TOU)*, *Real Time Pricing (RTP)*, *Critical Peak Pricing (CPP)* and *Variable Peak Pricing (VPP)* [9].

The power consumption of servers vary with the computing load on them [4], [15], [16]. So, by controlling the utilization of servers in a distributed system that comprises servers deployed at different geographical location, one could have control over the total electricity cost where the electricity prices are spatially and temporally different. However, the latency involved in servicing client requests by the remote servers may surpass the energy benefits if not carefully considered. The solution to the problem is far from trivial, given the large deviation among different electricity markets and time-varying end-to-end delays.

Keeping the electricity bill low in a scenario with redundant servers that are distributed across different regions while maintaining low response times is a challenging problem. As the U.S. electricity market varies greatly from region to region, the request routing problem must address the spatial-temporal feature of this system, such as the shift of peak hours of different time zones. We propose two learning approaches to control the customized web request routing. Both cases consider together the electricity cost and response time: (1) energy-delay aware with a S-model reward penalty epsilon (EDA-S) and (2) energy-delay aware with a P-model reward penalty epsilon (EDA-P). These algorithms exploit the spatio-temporal variations of electricity prices to reduce the operational costs. Our data confirms that it is possible to lower energy costs by up to 11% compared to common routing methods.

II. RELATED WORK

The literature presents numerous methods for distributing requests among multiple servers or Data Centers (DC) considering either delay or energy. We discuss two categories of related works here. Some works route requests to DCs at locations where the energy cost is cheaper. Other works consider the use of renewable energy jointly with spatial variations in price.

A. Load balancing by spatial pricing:

A two level technique to reduce the electricity cost in multi location DC was proposed [24]. At the first level, the spatial variations in electricity prices were exploited and at the second level scheduling strategies were proposed to distribute the requests among the heterogeneous servers at the selected DC. A detailed analysis of the U.S. electricity market was done with prices from 30 locations and a dynamic price-aware routing method was proposed showing 40% savings in simulation using Akamai's CDN traffic traces [20].

The stratus system used a Voronoi graph partition algorithm to determine DCs to service requests [7]. The service time, cost of the energy consumption and carbon emission were considered as metrics to be optimized in selecting a DC. A distributed cloud framework to reduce power consumption in the cloud based on dynamic speed scaling of processors was proposed [26]. Temporal and spatial variations in electricity prices were exploited to balance the load across the clusters and to reduce the energy consumption.

The total electricity cost reduction was formulated as a constrained mixed-integer problem and solved using Brenner's polynomial algorithm [22]. The energy consumption was optimized by global load balancing and at server level management by switching ON/OFF using mathematical evaluations [21]. A corrective marginal cost algorithm was proposed to solve the total energy cost by using a non linear integer programming optimization

model [30]. A Big Data analytic framework was proposed to minimize the total energy cost and delay for running big data analytic jobs on geographically distributed big data sets using a stochastic optimization method [34]. A recent work [33], proposes a solution to the cloud service providers who procure electricity from deregulated electricity market to optimize total electricity cost and bandwidth cost by dynamically routing the requests to DCs where electricity is cheaper.

B. Routing workloads to Renewable energy powered DCs:

A global load balancing of web requests to DCs where the renewable energy was maximum or the energy was cheaper was proposed [14]. Renewable energy was saved in Uninterrupted Power Supplies (UPS) and used during power fluctuations to reduce energy costs. The use of brown energy was reduced to zero by shifting the workloads to the DCs in regions where the energy cost was low and by switching OFF the servers at lightly loaded DCs [12].

A Fuzzy logic based load balancing method was proposed to route the workload among the geographically distributed Data Centers to reduce the electricity prices by considering the renewable energy availability, brown energy consumption and the average electricity cost at DC locations [27]. Methods of reactive global load balancing and auto scaling of servers were proposed to reduce the electricity cost and brown energy usage by routing the requests to sites where the renewable energy was higher and the cost was cheaper [28]. A distributed geographical load balancer was proposed [13], in which jobs were routed to DCs where the proportion of brown energy to total energy production was minimum. A cost function that comprises a linear combination of energy cost and lost revenue due to the SLA violation was used to route the web requests to a DC where the electricity price was minimum and powered by wind energy [6].

Our proposed approach provides per-request routing decisions whereas, the works that we surveyed offer a slow reaction to changes in the operational environment given that routing is only decided at the end of a time period (of order of minutes). This makes our method particularly suitable given the dynamics of current real-time pricing systems, which tend to work at much shorter time intervals. Another significant difference between our work and the current literature is that we provide experimental results from a network testbed instead of simulation.

III. PROBLEM STATEMENT

Consider an enterprise that has access to multiple Data Centers (DC) at different regions. Each location operates under the policy of the local Regional Transmission Organization (RTO). Let us assume that each location uses variable electricity pricing and that M_i servers participate in the application of interest at location i ,

$1 < i < N$. We consider the energy consumed by these servers, but exclude indirect consumption sources, such as lighting, cooling, and other servers. Let $p_j^i(t)$ be the average power in kW consumed by server j at DC i at time t obtained during the last second, that is, the average power of the interval between $t - 1$ and t . Consider $\gamma_i(t)$ to be the electricity rate at DC i at time t . The energy cost per hour calculated at time t is:

$$E^i(t) = \sum_{j=1}^{M_i} p_j^i(t) \gamma_i(t). \quad (1)$$

Here $\gamma_i(t)$ comprises the required conversion for kW to dollars/kWh. Long response times have a detrimental impact to the enterprise. We can associate this impact to a revenue loss. There are several possibilities available to define the revenue loss. For simplicity, we assume a linear model in this paper. The revenue loss increases with increase in delay. If d^i is the average response time at DC i and δ is the cost that we associate to delay, then the relative revenue loss at instance t is:

$$D^i(t) = \delta d^i(t) \quad (2)$$

The problem is to minimize the total cost (sum of the costs expressed by equations 1 and 2) incurred in delivering a requested service by routing the web request such that:

$$\min \left\{ \sum_{i=1}^N (kE^i(t) + (1-k)D^i(t)) \right\} \quad (3)$$

where k , $0 \leq k \leq 1$ is a constant that balances the importance of each factor (energy or delay) to the total cost. The choice of k is decided by the particular needs of the enterprise.

Iterative or direct methods are ill suited to deal with the highly dynamic nature of this system, which provide us the motivation to approach the problem with a learning method.

IV. LEARNING AUTOMATA

An automaton is defined as an entity that selects an action from a set of options to produce a favorable response at every instance. Learning is a process by which an entity will improve its behavior based on the knowledge acquired through self experience. LA learns to choose the best action on its own at every instance by observing the response from the environment through continuous interactions by implementing a feedback mechanism [18]. Figure 1 represents the interaction of stochastic LA with the environment. An automaton is formally described by a quintuple $\{\phi, s, X, G, A\}$. Where, X is the set of inputs i.e., $X \in \{0, 1\}$; ϕ is the set of internal states $\{\phi_1, \phi_2, \dots, \phi_n\}$; s is the output set $\{s_1, s_2, \dots, s_m\}$ with $m \leq n$; $G: \phi \rightarrow s$ denotes the output function and A is the algorithm that computes the state $\phi(n+1)$ from state

$\phi(n)$ depending on the response X it received from the environment.

The LA are classified as P-model, Q-model and S-model based on the responses it receives from the environment [18]. In P-model LA, the responses from the environment are mapped into binary values 0 and 1; In the Q-model, the responses are mapped into a finite number of values in the interval $[0, 1]$ and in S-model, they are mapped into continuous values in the interval $[0, 1]$. The most appropriate are the P-model and S-model for our problem. In both models, when the learning parameters a, b are equal, then they are known as P-model Reward Penalty LA and S-model Reward Penalty LA. When $b = 0$, they are known as P-model Reward Inactive LA and S-model Reward Inactive LA. When $b \ll a$, they are known as P-model Reward Penalty Epsilon (RPE) and S-model Reward Penalty Epsilon ($SRPE$). The parameters a and b are known as reward and penalty parameters respectively. In the beginning,

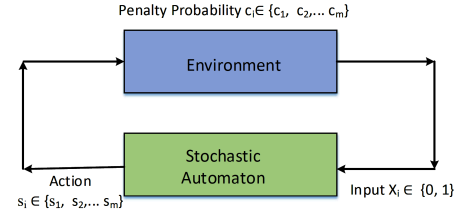


Figure 1: General operation of a stochastic learning automaton.

LA selects actions with equal probabilities which may not lead to optimal decisions. These decisions improve as the environment produces outputs that are used by the automaton in a feedback loop. For the environment, the input is the set $\{s_1, s_2, \dots, s_m\}$ and output is the set $X \in \{0, 1\}$, the probability $c_i \in \{c_1, c_2, \dots, c_m\}$ of producing unfavorable output $X = 0$ or $X = 1$ for the selected input/action is called as penalty probability and it is unknown, where $i = 1, 2, \dots, m$. The output function defines mapping of response from the environment to input to the LA. This function is different for P-model LA and S-model LA.

We define an LA that acts as a proxy server to route the client requests to the origin servers. Therefore, the action corresponds to a server selection whereas the environment's responses are the energy cost of the server and the response time of serving the the request. By observing both metrics through a feedback loop, the LA computes a cost function for the selected server and updates the selection probabilities according to a reinforcement scheme. In the P-model, if the value is less than the exponential average of cost values so far (favorable), it rewards the current server otherwise penalizes when the cost value is larger than the exponential average (unfavorable). In the S-model, the response of each of the server is recorded; the response of the

selected server is compared with the responses from others servers and the selected server is rewarded or penalized by a proportional amount.

It is usual in the web industry that proxy servers running from front end web portals redirect the requests to back end origin servers distributed across geographical regions. Our LA based proxy does the same but redirects to servers so as to optimize both energy cost and latency by a learning process. By using redundant proxies the redirection can be scaled up based on the intensity of the incoming requests.

V. ENERGY-DELAY AWARE ROUTING

For each arriving web request, an LA probabilistically selects a server v to process that request following the probability distribution Pr . Learning occurs by adjusting Pr upon completion of the transaction according to either the energy-delay aware with a S-model reward penalty epsilon (EDA-S) or the energy-delay aware with a P-model reward penalty epsilon (EDA-P) method. In both cases, the update occurs following an objective (or cost) function that is related to equation (3):

$$cost_{v,t} = k\gamma_i p_{v,t} + (1-k)\delta d_{v,t} \quad (4)$$

where v is the server providing the feedback at time t . Server v is hosted at DC i . Parameter γ_i is the electricity price at location i and δ is the loss of revenue associated with the request response time. Both $p_{v,t}$ and $d_{v,t}$ are the running averages of power and response time observed by the LA, which are balanced with parameter k .

Algorithms 1 and 2 describe EDA-S and EDA-P and are explained in the following subsections. The vector $prices$ contains the electricity rates of the N locations; The vector Pr contains the selections probabilities of the servers s_j , where $j = 1, 2, \dots, N$; The constants a, b are the learning parameters, and α is the exponential averaging parameter.

A. Energy-Delay Aware P-model Reward Penalty ϵ Server Selection (EDA-P)

After a service request is completed, the LA computes the delay and reads the power utilized by the server (v) that serviced the request. It then computes the cost associated with the server as $cost_v(t)$ and the exponential average cost $prev_cost(t)$. It then compares the current cost $cost_v(t)$ with $prev_cost(t)$. If $cost_v(t)$ is smaller, then the response is considered as favorable and the variable $reward$ is set to 1 else it is set to 0. When the next request is received by the LA and before making the decision, the LA updates the selection probabilities of servers based on the value of the variable $reward$. If the $reward$ is 1, the selection probability of the server that serviced the previous request is incremented and the probabilities of the other servers are decremented to make their sum equal to 1. Else if the $reward$ is 0, the probability of the selected server is reduced and the other probabilities adjusted.

Algorithm 1 EDA-P

```

1: Inputs:
    $prices = [\gamma_1, \gamma_2, \dots, \gamma_N]$ 
2: Initialize:
    $b \ll a < 1, 0 < \alpha, k < 1,$ 
    $Pr[s_j] = \frac{1}{N}, j = 1, \dots, N$ 
3: if  $http\_txn\_event$  is  $txn\_close$  then
4:    $cost_{v,t} \leftarrow k\gamma_i p_{v,t} + (1-k)\delta d_{v,t}$ , where  $v$  is the
     index of selected server
5:    $prev\_cost_t \leftarrow \alpha cost_{v,t} + (1-\alpha) prev\_cost_{t-1}$ 
6:    $reward = (cost_{v,t} < prev\_cost_t)$ 
7: else if  $http\_txn\_event$  is  $pre\_remap$  then
8:   if  $reward$  then
9:      $Pr[s_v] \leftarrow Pr[s_v] + a \times (1 - Pr[s_v])$ 
10:     $Pr[s_j] \leftarrow Pr[s_j] - a \times Pr[s_j] \forall j \neq v$ 
11:   else
12:      $Pr[s_v] \leftarrow (1-b) \times Pr[s_v]$ 
13:      $Pr[s_j] \leftarrow (1-b) \times Pr[s_j] + \frac{b}{N-1} \forall j \neq v$ 
14:   end if
15: end if

```

B. Energy-Delay Aware S-model Reward Penalty ϵ Server Selection (EDA-S)

The S-model differs from the P-model in the way it updates the action probabilities. The LA maintains values of $cost_{j,t}$ and $prev_cost_{j,t}$ for each server in separate vectors. It normalizes the response $prev_cost_{v,t}$ of a selected server with other servers by computing η to know how good or bad the response of the selected server compared to the other servers. It then increments (reward) or decrements (penalize) the probability of the selected server proportional to the response and adjusts the probabilities of other servers to make their sum equal to 1.

Algorithm 2 EDA-S

```

1: Inputs:
    $prices = [\gamma_1, \gamma_2, \dots, \gamma_N]$ 
2: Initialize:
    $b \ll a < 1; 0 < \alpha, k < 1$ 
    $Pr[s_j] = \frac{1}{N}; cost_j = [c_j]$ 
    $prev\_cost_j = [pc_j]$ , where  $j = 1, 2, \dots, N$ 
   and  $c_j, pc_j$  are some small integers
3: if  $http\_txn\_event$  is  $txn\_close$  then
4:    $cost_{v,t} \leftarrow k\gamma_i p_{v,t} + (1-k)\delta d_{v,t}$ , where  $v$  is the
     index of selected server
5:    $prev\_cost_{v,t} \leftarrow \alpha cost_{v,t} + (1-\alpha) prev\_cost_{v,t-1}$ 
6: else if  $http\_txn\_event$  is  $pre\_remap$  then
7:    $x_1 \leftarrow \min \{prev\_cost_j\} \forall j = 1, 2, \dots, N$ 
8:    $x_2 \leftarrow \max \{prev\_cost_j\} \forall j = 1, 2, \dots, N$ 
9:    $\eta \leftarrow \frac{prev\_cost_{v,t} - x_1}{x_2 - x_1}$ 
10:   $Pr[s_v] \leftarrow Pr[s_v] + (1-\eta)a(1 - Pr[s_v]) - b\eta Pr[s_v]$ 
11:   $Pr[s_j] \leftarrow Pr[s_j] - (1-\eta)a Pr[s_j] + b\eta(\frac{1}{N-1} - Pr[s_j])$ 
    $\forall j \neq v$ 
12: end if

```

VI. EXPERIMENTAL SETUP

We tested the approaches using an experimental network of four origin servers connected to two clients through a proxy server as in Figure 2. To make things simpler, we considered each server as located at a different DC (i.e., $M = 1$). The experimental test bed

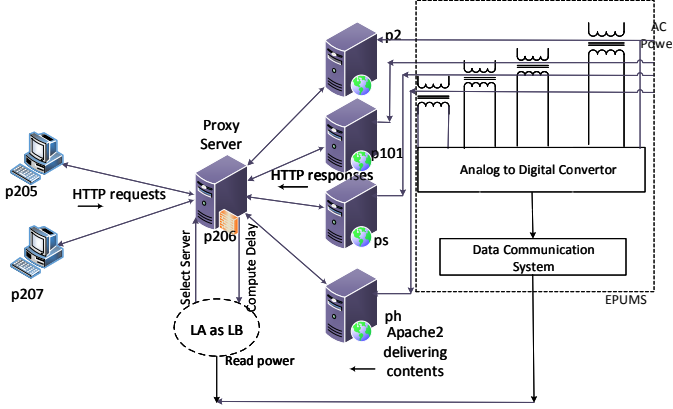


Figure 2: Testbed with hardware power sensors.

consists of PCs from our lab running on Linux operating system. The origin servers $p2$, $p101$, ps and ph are having different Energy Proportionality Index (EPI) and hold 4, 12, 2, and 2 cores respectively. The EPI is the measure of increase in power utilization of servers to the supplied load on them. The best EPI server shows less increase in power utilization to the increase in load. The servers are ranked according to the energy proportionality from best to worst are as follows: $p2$, $p101$, ps and ph and their power utilization are 0.2, 0.43, 0.39, 0.52 watts/request respectively. Figure 3 depicts the power usage of individual servers as a function of their utilization(load). The power usage is derived by observed power in watts normalized to peak power observed. The peak utilization is observed when the client sends the requests at rate of 180 requests/second. Even though server ph shows less increase in power with load, its idle power consumption is larger than $p2$. It is evident from the figure 3 that the power utilization of servers could be controlled by assigning proper amount of workload to them.

To emulate wide-area network delays, we inserted a random delay (normally distributed) to the links connecting origin servers and the proxy using “NetEm” tool. The links were rate limited with Token Bucket Filter (TBF) queues to emulate network congestion. The link connecting the origin server at node $p101$ was left with the original bandwidth i.e., with its full capacity of 1 GB and no delay is configured. The configured delays are summarized in Table I. Artificial delays are added to show that the algorithms are adaptive in optimizing latency in addition to energy cost.

The Apache2 server was installed on machines $p2$, $p101$, ps and ph to serve as web servers. The Apache

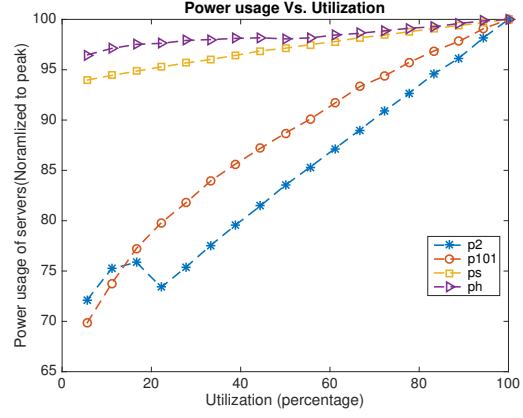


Figure 3: Power usage of servers increases with increase in supplied load

Table I: Configured rate limiting and delay

Link	TBF rate(Mbps)	Delay (msec)
$p2 \rightarrow P206$	100	8 ± 2
$ps \rightarrow P206$	100	10 ± 3
$ph \rightarrow P206$	20	20 ± 5

Traffic server 7.0.0 (ATS) was installed on $p206$ to act as the proxy server. It receives requests from the clients and forwards them to origin servers based on the remap strategy defined in the load balancer plug-in. It receives the contents from the origin servers and forwards back to the clients. It can cache the contents for fast access by clients. But we have disabled the caching to test the performance of the algorithms.

We have implemented our LA based energy-delay aware routing policies (and other request routing policies) in C++ and integrated it with the load balancer plug-in of ATS. The client program written in Python-2.7.12 sends HTTP requests for static files following a Poisson distribution from machines $p205$ and $p207$. The literature suggests the suitability of the Poisson traffic for modeling real traffic, which results from parallel packet arrivals from different sources [13]; packet arrivals at the Internet core follows a Poisson process [10], [5]; the request arrival process for individual documents on the web also exhibits a Poisson behavior [3]; interarrivals of user initiated sessions, such as TELNET and file transfer protocol (FTP), follow Poisson [19].

For each remapping policy, the requests are sent starting from $\lambda = 10$ requests/sec up to $\lambda = 90$ requests/sec in steps of $\lambda = 10$ requests/sec concurrently from both clients. For λ values larger than 180 requests/sec, the servers tend to respond with delays that exceeds traffic server’s time out values. When the incoming rate is larger than the processing speed of the servers, the requests are made to wait in the queue at the input ports of the servers. When the servers failed to respond before the timeout expires, the traffic server terminates the connections with origin servers and sends 502 status errors to clients. Therefore we have limited the maximum

value of λ to 180 requests/sec.

The Electrical Power Monitoring System (EPUMS) measures the power utilization by the origin servers and multicasts the values over the local area network [11]. A separate application written in C++ reads the power values and writes into a shared memory to be accessible by the LA. The LA reads the power values of all the servers before serving a request when the "pre_remap" event is fired at the traffic server. It then computes the energy cost by multiplying the power readings with γ , which includes the required conversion from Watts to kWh for the last 1 second duration along with the electricity rate. The delay is measured at the "transaction_close" event by measuring the time interval between the time at which client is connected to the proxy and the time at which the proxy has finished sending the file to the client for each request. The measured delay in seconds is multiplied with the constant δ to obtain the lost revenue. The cost function is updated with both computed energy cost from the server and the lost revenue due to the delay in serving the request by the server.

The power values are read from all the servers for each transaction even though the delay is obtained only from the selected server once a transaction has completed. The energy cost for all the servers are thus being updated at each transaction. The LA uses the cost values to update the selection probability of each server at the "pre_remap" event of the ATS. Based on the selection probabilities, the servers are selected at random to serve the next incoming request. The selection probability of a server at a location will be higher, where the price is lower and the requests are serviced with optimal latency until it's performance is better than the others. The selection probability of the optimal server will start decreasing and the probability of a next better performing server will start increasing when any of the following would happen: 1) The electricity price at other location starts cheaper, 2) The optimal server starts serving with increased delay comparing to other servers, 3) The energy cost at the location of optimal server increases either due to the increase in electricity rate or due to the increase in load. This process continues and servers are chosen to produce optimal performance in spite of the fluctuations in electricity prices and latency. The average delays and power utilization of servers are recorded at the client for plotting the results for each experiment.

VII. RESULTS

We have tested our cost-delay aware algorithms EDA-S and EDA-P in two different electricity price scenarios and compared with S-model Reward Penalty Epsilon (*SRPE*) and *RR* (Round Robin) algorithms.

The Round Robin is a common scheduling algorithm available in most proxy servers and most widely used method for distributing workload among the redundant

servers in the web industry due to its robustness, speed and simplicity [25], [28], [17]. Selecting servers based on the latency in receiving responses is proposed in [23], [32]. The *SRPE* method is a LA based routing algorithm which selects servers by optimizing latency. The implementation details of *SRPE*, could be found in our previous work [29].

We guarantee that our algorithms will work with dynamic pricing because the LA reads the power utilization by all the servers and compute the energy costs after each request. But we have used only one set of prices in both scenarios in this work due to time constraints. The pricing scenarios represent the fluctuations in the prices in real time electricity markets due to the fluctuations in demand, time zone differences, availability of renewable energy, use of different DR pricing such as time of day etc., for example, at one instance of time the servers at the proximity possess lower prices and at another instance of time the prices at distant DCs seems lower. The various parameters used in the algorithm are listed in Table II

Table II: Parameters

Parameter	Values	Purpose
α	0.9	proportion of current & past cost
k	{0.4, 0.6, 0.9}	proportion of energy & delay
$[a, b]$	[0.078, 0.006]	reward and penalty parameters

A. Pricing Scenario-I

In scenario-I, electricity prices were assigned as {4.70, 4.80, 14.47, 13.24} in cents/kWh at the locations of servers p_2 , p_{101} , p_s , p_h respectively. We have assumed that the servers are from DCs at different locations and each DC has one machine to make things simple. The prices are typical average electricity prices at different locations in the U.S. [1]. The Figures 4a–4d depict the performance of the cost-delay aware algorithms in terms of the electricity cost for a billing cycle of 30 days for four servers and the average delay evaluated with 10,000 samples for a range of demand values. In this scenario, server p_2 , which has the most efficient EPI, has an emulated delay of 8ms. The server p_{101} has more processing capabilities and does not have any delay on its link connecting to the proxy server. But its power consumption per request is more than the server p_2 . The EDA-P performed a bit worse than *SRPE*. EDA-S performed better in electricity cost for $k = 0.9$ and equal to *SRPE* in delay for $k = 0.4$.

The inferior performance of the EDA-P and EDA-S algorithms than *SRPE* is due to the fact that they are trying to optimize the costs involved with both energy and delay and utilized both p_2 and p_{101} servers in accordance with k . As p_2 has lost revenue associated with it due to the delay on its link, the costs involved with the above two algorithms are slightly more than *SRPE*. When $k = 0.9$, among the two servers, p_2 has been selected more number of times than p_{101} and

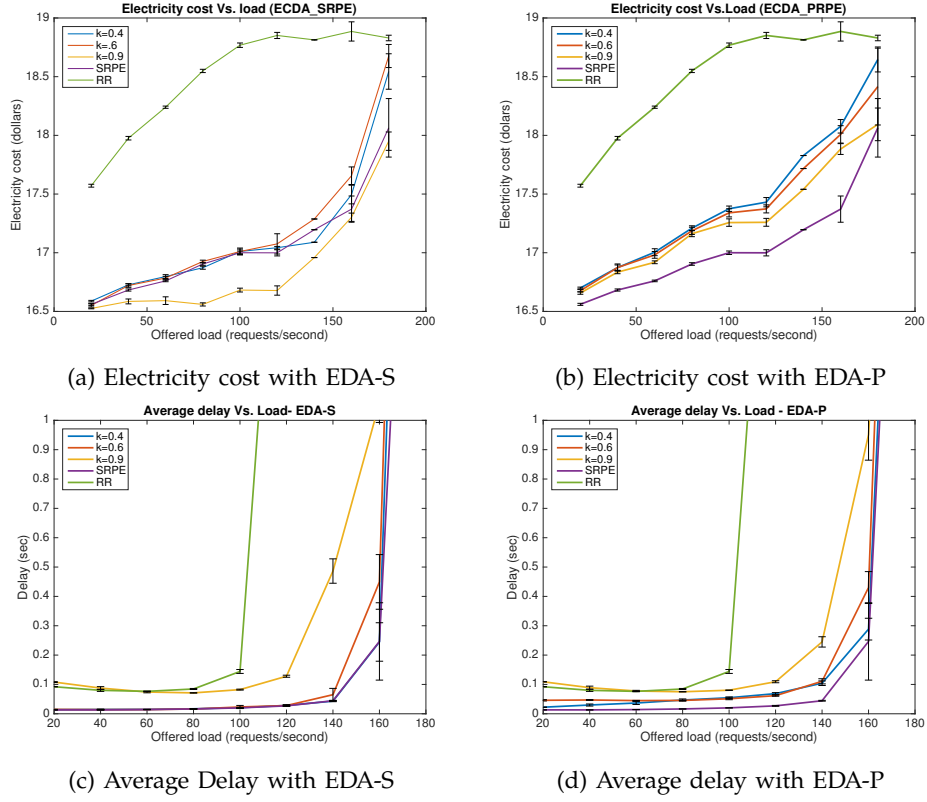
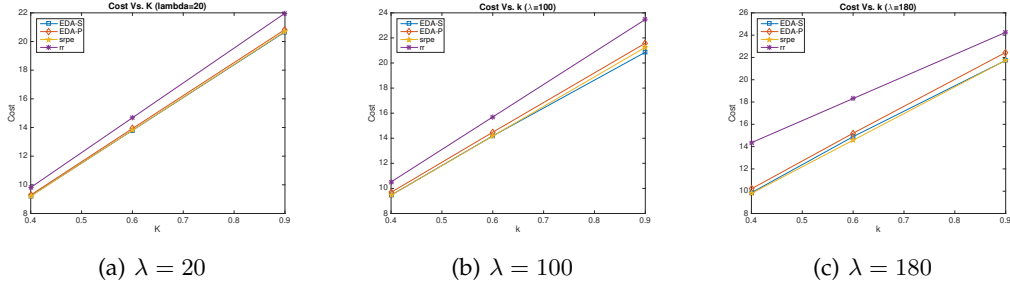


Figure 4: Performance of Energy-cost and delay aware algorithms (scenario-I)

Figure 5: Cost vs. k (Scenario-I). Parameter k is the weight factor of energy in the cost function.

reverse for $k = 0.4$. But as the *SRPE* algorithm just selects a server which gives the minimum delay, it selects p_{101} most number of times and as it does not have any delay, the lost revenue associated with it is low.

The advantage of the energy aware algorithms could be better realized in scenario-II. We have plotted all our results with 95% confidence intervals.

The graphs in Figure 5 demonstrate the influence of the parameter k on the resulting combined cost in pricing scenario-I (energy cost and lost revenue) for λ values of 20 requests/sec, 100 requests/sec and 180 requests/sec. It is evident that for all values of k , the LA based algorithms have shown better cost savings. The advantage of using LA based algorithms are better visible for larger values of λ . When λ values are higher, both power consumption and energy cost will be higher

and as the LA algorithms dynamically selects servers with optimal energy costs, the total average energy costs are lower than using RR.

B. Pricing Scenario-II

In scenario-II, the prices were assigned to servers in the following order: $\{13.24, 14.47, 4.80, 4.70\}$ such that the server with small delay, higher processing capabilities and better EPI are having higher electricity prices. Figures 6a–6d depict the performance of our algorithms in scenario-II. It can be observed that the cost-delay aware algorithms performed better than *SRPE* in terms of electricity cost for all values of k , but performed worse in terms of average delay because the servers with higher emulated delay have lower energy costs. So our algorithms tried to optimize the combined cost

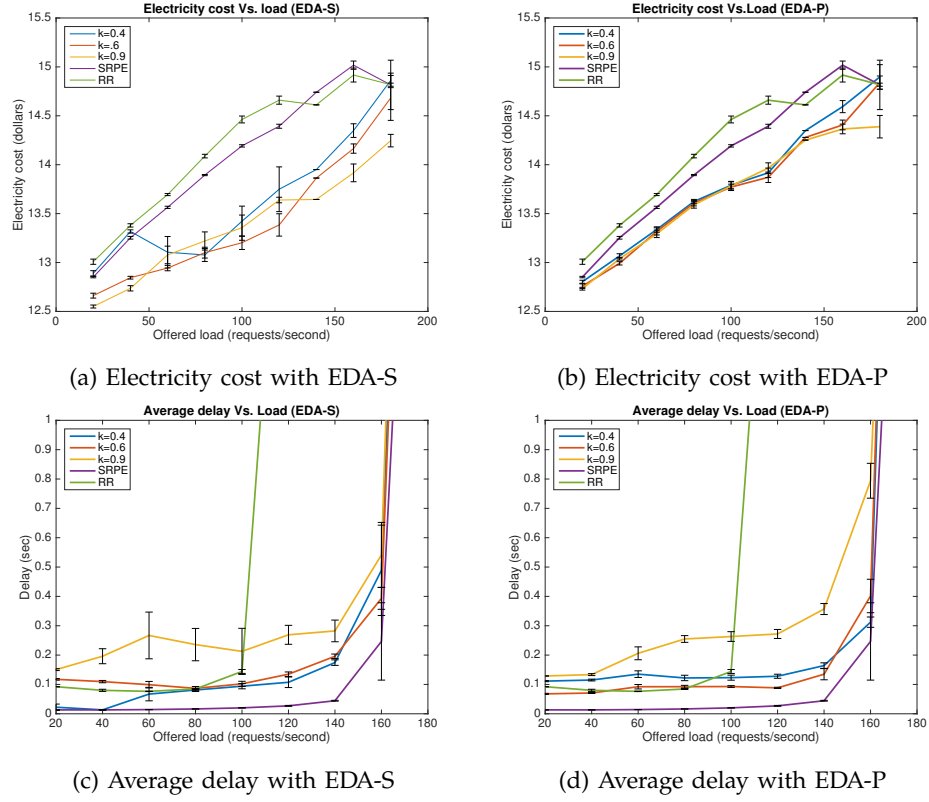
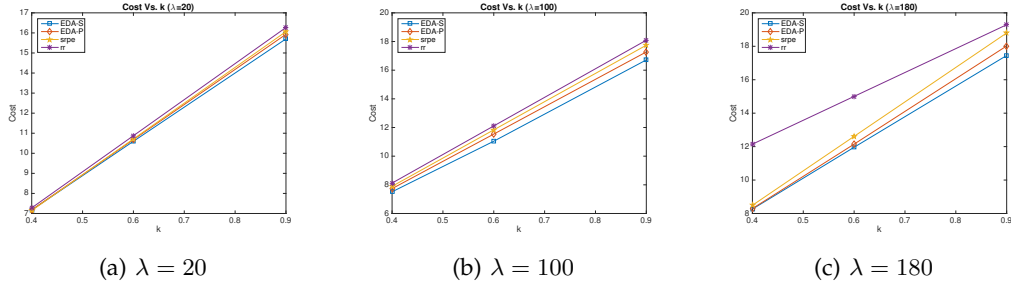


Figure 6: Performance of Energy-cost and delay aware algorithms (scenario-II)

Figure 7: Cost vs. k (Scenario-II). Parameter k is the weight factor of energy in the cost function.

with acceptable services delays based on the k values. Both algorithms performed better than round robin (RR) method in terms of both energy cost and delay. The graphs in Figure 7 demonstrate the influence of the parameter k in the resulting combined cost in the pricing scenario-II (energy cost and lost revenue). The energy cost and delay aware algorithms provide a control knob through the parameter k by which we could impose the importance we want to give for energy price and latency.

VIII. CONCLUSION

We have introduced two self-adapting request routing methods that exploit learning to distribute client requests among geographically distributed servers to achieve low electricity costs and delays. In scenario-I, when $k = 0.9$ and requests/sec is 120, EDA-S reduces 11.53% of the

electricity cost compared to RR and EDA-P brings down costs by 8.44% than RR. EDA-S shows 2% lower cost than SRPE for 80 requests/sec. In scenario-II, when the demand is at 120 req/s and $k = 0.6$, EDA-S shows 8.6% and EDA-P 5.3% lower electricity costs than RR. EDA-S shows 7.2% lower cost than SRPE at 160 req/s and at $k = 0.9$ and EDA-P shows a 4.3% reduction. The proposed algorithms can optimize costs even when the electricity costs are time varying, such as in demand-response programs and can also be customized to optimize delay for latency sensitive applications. Our future work will test the impact of time varying prices and larger number of servers with real web traffic traces.

ACKNOWLEDGMENT

The authors would like to thank Dr. Jaspal Subhlok and Dr. Omprakash Gnawali for their valuable comments on this work. This work was supported by an Early Career Faculty grant from NASA's Space Technology Research Grants Program.

REFERENCES

- [1] E. I. Administration, "Electric power monthly," https://www.eia.gov/electricity/monthly/epm_table_grapher.php?t=epmt_5_6_a, accessed:2017-10-10.
- [2] M. H. Albadi and E. F. El-Saadany, "A summary of demand response in electricity markets," *Electric power systems research*, vol. 78, no. 11, pp. 1989–1996, 2008.
- [3] M. F. Arlitt and C. L. Williamson, "Internet web servers: Workload characterization and performance implications," *IEEE/ACM Transactions on networking*, vol. 5, no. 5, pp. 631–645, 1997.
- [4] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, 2007.
- [5] J. Cao, W. S. Cleveland, D. Lin, and D. X. Sun, "Internet traffic tends toward poisson and independent as the load increases," in *Nonlinear estimation and classification*. Springer, 2003, pp. 83–109.
- [6] H. Dou, Y. Qi, W. Wei, and H. Song, "A two-time-scale load balancing framework for minimizing electricity bills of internet data centers," *Personal and Ubiquitous Computing*, vol. 20, no. 5, pp. 681–693, 2016.
- [7] J. Doyle, R. Shorten, and D. O'Mahony, "Stratus: Load balancing the cloud for carbon emissions control," *IEEE Transactions on Cloud Computing*, vol. 1, no. 1, pp. 1–1, 2013.
- [8] X. Fang, S. Misra, G. Xue, and D. Yang, "Smart grid: The new and improved power grid: A survey," *IEEE communications surveys & tutorials*, vol. 14, no. 4, pp. 944–980, 2012.
- [9] P. C. Jennifer Potter, "Demand response advanced controls framework and assessment of enabling technology costs," https://emp.lbl.gov/sites/default/files/demand_response_advanced_controls_framework_and_cost_assessment_final_published.pdf, accessed:2017-12-18.
- [10] T. Karagiannis, M. Molle, and M. Faloutsos, "Long-range dependence ten years of internet traffic modeling," *IEEE internet computing*, vol. 8, no. 5, pp. 57–64, 2004.
- [11] R. Lent, "A sensor network to profile the electrical power consumption of computer networks," in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*. IEEE, 2010, pp. 1433–1437.
- [12] M. Lin, Z. Liu, A. Wierman, and L. L. Andrew, "Online algorithms for geographical load balancing," in *Green Computing Conference (IGCC), 2012 International*. IEEE, 2012, pp. 1–10.
- [13] Z. Liu, M. Lin, A. Wierman, S. Low, and L. L. Andrew, "Greening geographical load balancing," *IEEE/ACM Transactions on Networking (TON)*, vol. 23, no. 2, pp. 657–671, 2015.
- [14] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. Andrew, "Geographical load balancing with renewables," *ACM SIGMETRICS Performance Evaluation Review*, vol. 39, no. 3, pp. 62–66, 2011.
- [15] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "A power benchmarking framework for network devices," *NET-WORKING 2009*, pp. 795–808, 2009.
- [16] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: eliminating server idle power," in *ACM sigplan notices*, vol. 44, no. 3. ACM, 2009, pp. 205–216.
- [17] M. Mesbahi and A. M. Rahmani, "Load balancing in cloud computing: a state of the art survey," *International Journal of Modern Education and Computer Science*, vol. 8, no. 3, p. 64, 2016.
- [18] K. S. Narendra and M. A. Thathachar, *Learning automata: an introduction*. Courier Corporation, 2012.
- [19] V. Paxson and S. Floyd, "Wide area traffic: the failure of poisson modeling," *IEEE/ACM Transactions on Networking (ToN)*, vol. 3, no. 3, pp. 226–244, 1995.
- [20] A. Qureshi, R. Weber, H. Balakrishnan, J. Gutttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," in *ACM SIGCOMM computer communication review*, vol. 39, no. 4. ACM, 2009, pp. 123–134.
- [21] L. Rao, X. Liu, M. D. Ilic, and J. Liu, "Distributed coordination of internet data centers under multiregional electricity markets," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 269–282, 2012.
- [22] L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.
- [23] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 2002, pp. 1190–1199.
- [24] A. N. Sankaranarayanan, S. Sharangi, and A. Fedorova, "Global cost diversity aware dispatch algorithm for heterogeneous data centers," in *ACM SIGSOFT Software Engineering Notes*, vol. 36, no. 5. ACM, 2011, pp. 289–294.
- [25] M. Slocombe, M. Miller, C. Ajalat, and V. Fuller, "Content request routing and load balancing for content distribution networks," Apr. 24 2003, uS Patent App. 09/982,721.
- [26] R. Stanojevic and R. Shorten, "Distributed dynamic speed scaling," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–5.
- [27] A. N. Toosi and R. Buyya, "A fuzzy logic-based controller for cost and energy efficient load balancing in geo-distributed data centers," in *Utility and Cloud Computing (UCC), 2015 IEEE/ACM 8th International Conference on*. IEEE, 2015, pp. 186–194.
- [28] A. N. Toosi, C. Qu, M. D. de Assunção, and R. Buyya, "Renewable-aware geographical load balancing of web applications for sustainable data centers," *Journal of Network and Computer Applications*, vol. 83, pp. 155–168, 2017.
- [29] G. Velusamy and R. Lent, "Smart load-balancer for web applications," in *Proceedings of the 2017 International Conference on Smart Digital Environment*. ACM, 2017, pp. 19–26.
- [30] P. Wang, L. Rao, X. Liu, and Y. Q. D-pro, "dynamic data center operations with demand-responsive electricity prices in smart grid. smart grid," *IEEE Transactions on*, vol. 3, no. 4, pp. 1743–1754, 2012.
- [31] J. Whitney and P. Delforge, "Data center efficiency assessment," *Issue paper on NRDC (The Natural Resource Defense Council)*, 2014.
- [32] E. W. Zegura, M. H. Ammar, Z. Fei, and S. Bhattacharjee, "Application-layer anycasting: A server selection architecture and use in a replicated web service," *IEEE/ACM Transactions on Networking (TON)*, vol. 8, no. 4, pp. 455–466, 2000.
- [33] Y. Zhang, L. Deng, M. Chen, and P. Wang, "Joint bidding and geographical load balancing for datacenters: Is uncertainty a blessing or a curse?" in *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE, 2017, pp. 1–9.
- [34] P. Zhao, S. Yang, X. Yang, W. Yu, and J. Lin, "Energy-efficient geodistributed big data analytics," *arXiv preprint arXiv:1708.03184*, 2017.