Renewable-aware Geographical Load Balancing of Web Applications for Sustainable Data Centers

Adel Nadjaran Toosi[a], Chenhao Qu[a], Marcos Dias de Assunção[b], Rajkumar Buyya[a]

[a] *Cloud Computing and Distributed Systems Laboratory, School of Computing and Information Systems, The University of Melbourne, Australia*
[b] *Inria, ENS de Lyon, France*

Abstract

The ever-increasing demand for web applications deployed across multiple data centers results in large electricity costs for service providers and significant impact on the environment. This has motivated service providers to move towards more sustainable data centers powered by renewable or green sources of energy, such as solar or wind. However, efficient utilization of green energy to service web applications is a challenging problem due to intermittency and unpredictability of both application workload and renewable energy availability. One possible solution to reduce cost and increase renewable energy utilization is to exploit the spatio-temporal variations in on-site power and grid power prices by balancing the load among multiple data centers geographically distributed. In this paper, we propose a framework for reactive load balancing of web application requests among Geo-distributed sustainable data centers based on the availability of renewable energy sources on each site. A system prototype is developed, its underlying design and algorithms are described, and experiments are conducted with it using real infrastructure (Grid'5000 in France) and workload traces (real traffic to English Wikipedia). The experimental results demonstrate that our approach can reduce cost and brown energy usage with efficient utilization of green energy and without a priori knowledge of future workload, availability of

Email addresses: anadjaran@unimelb.edu.au (Adel Nadjaran Toosi), cqu@student.unimelb.edu.au (Chenhao Qu), marcos.dias.de.assuncao@ens-lyon.fr (Marcos Dias de Assunção), rbuyya@unimelb.edu.au (Rajkumar Buyya)

renewable energy, and grid electricity prices.

1. Introduction

Data centers are known to be consuming enormous amount of power leading to high operational cost and high carbon footprint on the environment. According to a report from NRDC[1] [1], in 2013, US data centers alone consumed 91 billion kilowatt-hours of electricity, equivalent to two-year power consumption of all households in New York city. This is projected to increase to roughly 140 billion kilowatt-hours and is responsible for the emission of nearly 150 million metric tons of carbon dioxide per annum in 2020. These costs and environmental concerns have prompted service providers to reduce their energy consumption and their dependence on power generated from fossil fuels (i.e., Brown energy).

Large companies (e.g., Google[2], Microsoft[3] and Amazon[4]) are working towards sustainable data centers by using renewable energy sources (i.e., Green energy) and making direct investments in on-site green power generation. *Photovoltaic solar panels* that directly convert sunlight into electricity and *wind turbines* that capture wind energy and turn it into electricity are among the most popular on-site power sources used by contemporary data centers. For example Amazon Web Services (AWS) is building a wind farm that will be operational by late 2016 and generate 40 percent of its electrical usage.[5]

Powering data centers entirely with renewable energy sources, unlike brown energy, is challenging due to the intermittent and unpredictable availability of wind and solar energy. For example, photovoltaic (PV) solar energy is only available during the day time and the amount of power produced depends on the weather and geographical location of the data center. To mitigate this variability, besides on-site renewable energy sources, service providers end up using grid power or brown energy as a backup in their data centers. However, to minimize brown energy usage, they need to obtain the highest possible renewable energy utilization.

[1] Natural Resources Defense Council, `www.nrdc.org`.

[2] `http://www.google.com.au/green/energy/`.

[3] `http://www.microsoft.com/environment/renewable.aspx/`.

[4] `http://aws.amazon.com/about-aws/sustainable-energy/`.

[5] `http://www.reuters.com/article/2015/07/14/us-amazon-iberdrola-idUSKCN0PO1PF20150714`.

Even though data centers can store power generated by renewable sources of energy in batteries for later needs, this approach has many problems [2]. For example, 1) batteries lose energy due to internal resistance and self-discharge, 2) battery-related costs can dominate the cost of power systems, and lastly 3) batteries use chemicals that are harmful to the environment. Given aforementioned problems, the best way to take full advantage of the available green energy is to match the energy demand to supply.

There is a large number of studies illustrating the potential of using "*geographical load balancing*" in reducing brown energy usage and accordingly maximizing renewable energy utilization [3, 4, 5]. Geographical load balancing (GLB) allows for "*follow the renewables*" by utilizing resources from geographically distributed data centers [4]. Additionally, it routes the load to places with lower electricity prices even if renewable power is fully utilized or not available. This eventually leads to significant cost savings.

Among different types of applications, web-applications are highly popular and widely adopted these days. Web applications are ideal for geographical load balancing as they can quickly adapt to changes in the demand and their mostly small sized requests can be easily redirected among multiple data centers. Using multi-tier clustered web server architectures, web applications are able to efficiently allocate resources within and among data centers according to time varying demand and renewable energy availability. In fact, if the Quality of Service (QoS) requirement of web requests in terms of response time can be satisfied, the load balancing algorithm distributes requests among targeted data centers so that overall renewable energy usage is maximized. Please note that web application workload is not deferrable that means every time a request is received, the response should be generated immediately afterwards. Therefore, the load balancing technique must make real-time scheduling of the load which does not delay the current requests.

Figure 1 depicts an architectural overview of multiple sustainable data centers and a green load balancer. The *power infrastructure* generates and delivers power for the IT equipment and cooling facilities of the data center using grid power and local renewable power supplies. Data centers host resources for the web application and the *load balancer* is responsible for sharing requests according to the availability of renewable energy sources on each site. In this paper we aim to address the interesting and challenging question posed by this architecture: "Without a priori knowledge of the future demand, dynamic and unpredictable nature of renewable energy sources, and electricity prices, how can the load balancer distribute web application

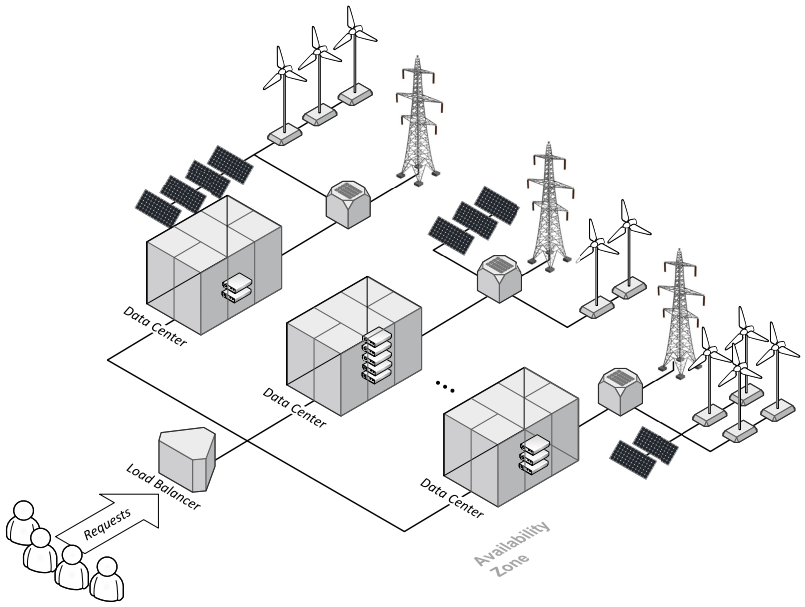


Figure 1: Architectural overview of sustainable data centers and green load balancing.

requests among multiple data centers so that the overall renewable energy usage is maximized and the total cost of power consumption is minimized?"

To address this problem, we propose a framework for reactive load balancing of web application requests among multiple geographically distributed data centers based on the availability of renewable energy sources on each site. We develop a prototype, detail its underlying design and algorithms, describe technical aspects of that, and experiment with it using real life infrastructure (i.e., Grid'5000 in France) and workload traces (i.e., real traffic to English Wikipedia). We also model renewable energy power generation using real traces of meteorological data for wind and solar radiation in the location of each data center. Using the prototype, we evaluate the optimization techniques and demonstrate that our simple, yet practical, approach can achieve significant cost savings without advance knowledge of future demands, availability of renewable energy, and electricity prices. The proposed load balancing algorithm is triggered periodically and collects the current available renewable power and electricity price at each data center. Then, based on the present rate of requests at the load balancer, it adjusts the load distribution among data centers. By using this technique, load distribution among data centers is adapted to the dynamic and varying renewable power

and electricity prices.

The load balancing technique proposed in this paper is an *online algorithm* that acts without future knowledge of demand, renewable energy availabilities, and electricity prices. Online algorithms have been previously used in the literature to tackle the problem of geographical load balancing [6, 5]. However, since we focus on the implementation aspects of the system, we do not provide the analytical reasoning on the *competitive analysis* of the proposed online algorithm.

One of the unique features of our work is that in contrast to the majority of other studies evaluating their system performance through simulations and analytical reasoning; our experiments are conducted in a real testbed with realistic workload traces. Moreover, results of experiments are generated based on the fine-grained measurements of power consumption using real-time probes provided by a live monitoring system.

The **main contributions** of the paper are as follows:

- A reactive load balancing algorithm to distribute a web application load among different data centers in a region to maximize on-site renewable energy utilization at each data center and to minimize overall cost. Our proposed method has a linear order of complexity and does not require any future knowledge of demands, availability of renewable energy, and electricity price. This removes the need for any prediction component and its simplicity is very appealing in practice.

- Design and implementation of a two-layered load balancing prototype by extending "3-tier architecture" common to web applications.

- Evaluation and validation of the proposed load balancing algorithm using the developed system prototype in a real testbed with realistic workload traces. Results of experiments are generated based on the real-time measurements of actual power consumption. Meteorological data in the location of each data center used to model renewable power generation.

The rest of the paper is organized as follows: Section 2.1 motivates our work. We describe the system architecture in Section 2.2. Section 2.3 presents a detailed discussion on the design and implementation of the system. Section 3 proposes our load balancing and auto-scaling algorithms. The performance evaluation of the system is presented in Section 4. First,

we describe the testbed setup, the benchmark algorithm, traces of workload, renewable power and electricity prices in Section 4.1. Then, benchmark policies and the experimental results are discussed in Sections 4.2 and 4.3, respectively. Section 5 discusses related work. Finally, our conclusions are presented in Section 6.

2. Load balancing in sustainable data centers

2.1. Motivation and background

Internet-scale distributed systems such as web applications use resources provided by geographically dispersed data centers each with hundreds of thousands of physical nodes. The enormous and growing energy demands of such data centers have motivated constructing sustainable data centers for both economic and environmental reasons. Renewable sources of energy (e.g, wind and solar) have the potential to play an important role in such sustainable data centers. However, the intermittent and variable nature of renewable energy sources, caused by being heavily dependent on weather conditions, prevents them from being used as primary power supplies for data centers. Essentially, a data center must be operational even when renewable energy is not available. This is possible by feeding the data center with brown energy from the grid in case the local renewable energy is insufficient. Data centers often contract with power companies to pay variable brown electricity prices for their excess usage provided by the utility grid.

Web application providers using resources from sustainable data centers (i.e., those equipped with on-site renewable energy facilities in our definition) in a specific *region* can distribute load by following the renewable energy supply in each site to save cost and reduce carbon emissions. That is, more requests must be redirected to the places with higher availability of renewable energy. The availability of the renewable electricity at each data center might vary due to reasons such as geographical location, weather conditions, workload, and capacity of power generators. Eventually, if there is not enough available renewable power from all data centers to handle the entire load, requests must be routed to places with lower electricity price. Indeed, with widespread adoption of smart grid technologies, spatial and temporal differences in electricity prices, even in a small region, provide an opportunity for the load redirection to save cost.

In order to make this possible, we design and implement a framework for green load balancing of web requests among sustainable data centers host-

ing the web application and dispersed throughout a region. In this system, it is assumed that the "operational cost" of on-site power generated from renewable sources of energy at each data center is zero. This is a valid assumption as renewable power generation needs one-time installation and very low maintenance cost during the lifespan of the renewable power generators. Decision making regarding the investment on on-site renewable power generation based on its "capital cost" and "return on investment" is not in the scope of this work. Interested readers are referred to [7] for feasibility of powering internet-scale systems using renewable energy and optimal portfolio of solar and wind energy mix.

Moreover, delay related changes due to geographical routing are not a concern in this work as we only consider data centers located within a certain region. In our terminology, a region is a geographic area (e.g., Europe, US East, Central Asia) with multiple and isolated data centers connected through low latency links. In the performance evaluation section, we show that for a testbed of the same characteristics, response time remains below the acceptable delay requirements of the studied web application (i.e., Wikipedia)[6] and will not significantly be affected due to geographical load balancing within the region.

2.2. System Architecture

Our approach to green load balancing of web applications extends multi-tier clustered web server architecture by adding an extra layer of load balancing responsible for sharing load at the data-center level as shown in Figure 2. A standard multi-tier web application often consists of three logical layers [8]:

1. Presentation Layer – represents the interface displayed to the end-user, e.g., a web page viewed on a web browser.
2. Business/Domain Layer – implements the core application logic, e.g., core web application deployed on web servers.
3. Data Layer – handles access to the persistent storage, e.g., a database server.

This layered architecture allows for software components of each layer to be deployed within single/multiple separate machines and easily scale out based on the load. For example, the number of *web servers* can increase or decrease

[6]Web-based encyclopedia project supported by the Wikimedia Foundation.

dynamically in response to the demand. Web servers of the business/domain layer are often deployed behind a *load balancer*, which redirects the incoming requests among them. In our architecture, we introduce two level of load balancing:

1. *Local Load Balancing* – redirects requests between web servers within a data center, and
2. *Global Load Balancing* – redirects requests among local load balancers, each associated with a data center.

In this paper, we focus on load sharing at the *global load balancer* (Global-LB). Global-LB is the main entry point of the system and all incoming requests at this point are distributed among geographically dispersed data centers based on the proposed policy.

In each data center, there is an *auto-scaler* that dynamically and adaptively adds and removes web servers behind the local load balancer in response to the dynamic workload. Here, we only focus on auto-scaling of the application layer which is a common practice in the real world. Auto-scaling can be done based on statistics gathered by the monitors responsible for fetching corresponding system information such as resource utilization at web server nodes, request rates, etc. According to the obtained information, when it is necessary, the auto-scaler then makes scaling decisions based on predefined strategies and policies. In Section 2.3.2, we detail our proposed auto-scaling policy.

At the lowest level of the system architecture, Database server(s) provide access to the persistent storage for the web servers. Web servers in the business/domain layer query the database servers of the data layer for required information (e.g., web page content). The data layer can be composed of both transactional relational databases, caches and novel approaches like NoSQL and NewSQL databases.

2.3. Design and Implementation

Apart from components of multi-tier architecture for web applications, the key components of the proposed system are *global load balancer* and *autoscaler*.

2.3.1. Global load balancer (Global-LB)

The function of Global-LB is to redirect incoming web requests to an appropriate data center site so that the overall renewable energy utilization

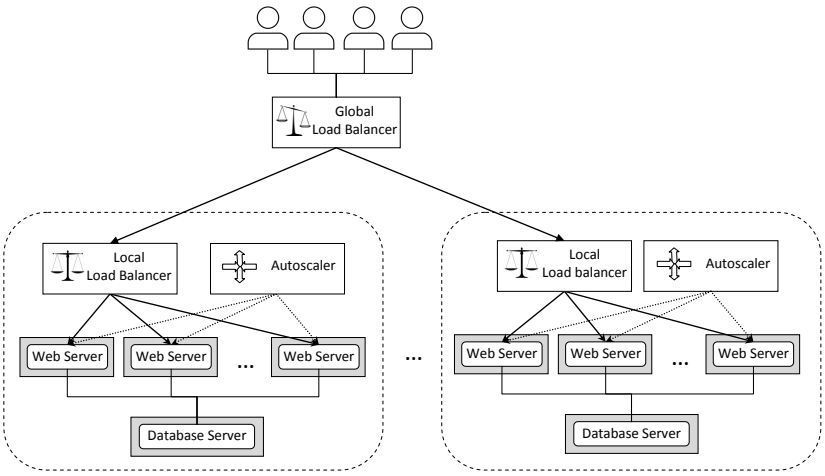


Figure 2: Overall system architecture and related components.

is maximized. To achieve this goal, we use HAProxy[7] load balancer that distributes requests across local load balancers. The "weighted round robin" is one of the main load balancing algorithms used by HAProxy to determine which server, in the backend, is selected for the next incoming request. A weight parameter can be assigned to each server in the backend to manipulate how frequently the server is selected for the request routing, compared to other servers.

We designed and developed a *controller* to be run besides HAProxy, as shown in Figure 3. The role of the controller, which is a Java program deployable on the same or separate host as HAProxy, is to assign weights to each server on a regular basis according to the current status of the system. In order to do so, the controller has a *decision making module* calculating weights based on the load balancing policies and information collected by *monitoring modules*. Monitoring modules are responsible for computing the amount of power consumed on each site and the total number of requests redirected to the site in a certain time window. To obtain the number of requests in a time period, the monitoring module queries the statistics measured by HAProxy. To compute the power consumption, the corresponding monitoring module communicates with the power monitoring APIs provided

[7]HAProxy - The Reliable, High Performance TCP/HTTP Load Balancer, `http://www.haproxy.org/`.

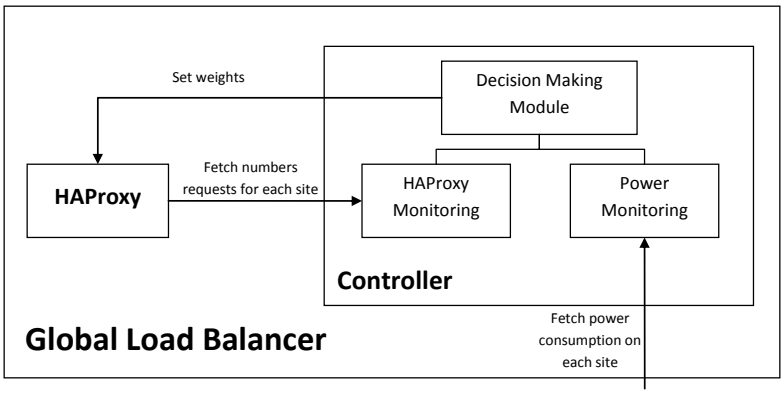


Figure 3: Global Load Balancer.

on each data center site. In our prototype, the monitoring module directly fetches the power consumption from every node using probes installed on the node. Having collected the required information, the decision making module based on the predefined load balancing policy, which will be discussed shortly, sets the weight values relative to each site for HAProxy. The load balancing policy used in our system is presented in Section 3.1.

2.3.2. Auto-scaler

Each *auto-scaler* is packaged in a single JAR file and deployed on the same or a separate host as the local load balancer. The main function of an auto-scaler is to horizontally scale out web servers on demand (to allocate or deallocate web server machines) in the business/domain layer of the web application.

There is a substantial amount of work on designing auto-scaling solutions for multi-tier applications, for example [9, 10, 11]. Among different approaches, threshold-based auto-scaling methods that work based on performance metrics such as CPU and RAM utilization are among the most widely adopted techniques, e.g., AWS Auto-Scaling Service.[8] More resources are provisioned whenever an upper threshold is exceeded and resources are released whenever a lower threshold is reached. The main scope of this work is not to propose a new methodology of auto-scaling for multi-tier web applications. Any form of auto-scaling method can be plugged into the system

[8]Auto-Scaling - Amazon Web Services, `http://aws.amazon.com/autoscaling/`.

without the necessity to modify other components. Nevertheless, in our prototype, we need an efficient auto-scaling mechanism to elastically scale out the web server machines in the virtual cluster once it is required.

Since in our evaluation we use homogeneous server farm[9] for web servers in each data center, we employ a simple yet effective auto-scaling method based on profiling data collected from web server machines in each data center. Accordingly, we set an upper threshold for the request rate at or below which web server machines of the specific type can provide responses in a timely manner. Thus, the auto-scaler allocates more web server machines whenever request rate goes higher than the threshold. In section 3.2, we present a detailed discussion on the auto-scaling algorithm.

3. Load Balancing and Auto-Scaling Policies

In this section, first, we detail our proposed load balancing algorithm used in Global-LB and then the auto-scaling algorithm employed to scale out web servers in each data centers is presented.

3.1. Green Load Balancing Policy

Algorithm 1 shows the pseudo code of the Green Load Balancing (GreenLB) policy used by the Global-LB component of the proposed system. Variable R is defined to maintain the overall rate of requests that can be handled by the power generated from renewable energy sources in all data centers. Lines 2–9 gather the required information using the monitoring modules and compute request rates at which a data center could accommodate requests only using renewable power within the time window (e.g., for the last 10 minutes). c and t keep track of the energy consumption and total number of requests served by the data center, respectively (Lines 3 and 4). a is a value for the total amount of renewable power currently available in the site (Line 5). By dividing value of c to t, we compute the amount energy w consumed at the specific data center to serve each request (Line 6). Accordingly, the maximum rate of requests r_d specifying the rate at which this data center can provide service only using renewable power (Line 7) is computed. The value of R is updated based on the calculated rate r_d iteratively.

When the computation of r_d is done for each data center, the request rate at which global load balancer received requests in the recent time window, γ,

[9] A server farm made up of machines having the same characteristics.

Algorithm 1 Green Load Balancing (GreenLB) Policy

$R \leftarrow 0$
for all data centers d in the list **do**
    $c \leftarrow$ Fetch the data center's energy consumption in Watt-hour within the time window
    $t \leftarrow$ Fetch the number of requests redirected to the site within the same time window
    $a \leftarrow$ Fetch currently available renewable power at the site in Watt
    $w \leftarrow$ Compute Watt-hour consumption per request $(c \div t)$
    $r_d \leftarrow$ Compute the request rate (#reqs/hour) data center d can accommodate using renewables $(a \div w)$
    $R \leftarrow R + r_d$
end for
$\gamma \leftarrow$ Fetch request rate (#reqs/hour) at Global-LB
if $\gamma < R$ **then**
    for all data centers d in the list **do**
        set weight as $r_d \div R$
    end for
else
    Find the data center d' with the cheapest price of brown energy per request.
    $L \leftarrow \gamma$
    for all data centers d in the list except d' **do**
        set weight as $r_d \div \gamma$
        $L \leftarrow L - r$
    end for
    Set the weight for d' as $L \div \gamma$
end if
Update HAProxy weights accordingly

is fetched from HAProxy (Line 10). If $\gamma < R$, it means that enough renewable energy is available to handle all incoming requests at the current rate (Line 11) and the weight for each data center is computed proportionally to the availability of renewable energy on each site (Line 13). Otherwise, if there is not enough renewable energy available to handle all requests at this rate (Line 15), each data center will receive requests based on the maximum rate of requests they can serve with renewable energy sources (Line 19) and the one with the lowest price of brown energy consumption per request accommodates the remaining parts of requests (Line 22). Finally, at Line 24, the HAProxy weight parameters are updated according to the computed values.

Please note that our policy never deallocates all web server machines in a specific data center and there is always at least one web server running on each data center. Therefore, if the availability of renewable energy sources at a certain data center is lower than the energy required for running even one web server machine, we set the weight in a way that the data center receives the minimum rate enough for one web server. Details related to this calculation are not shown in Algorithm 1 for the sake of preserving clarity. In addition, we assumed that there are sufficient resources available in each data center to accommodate the whole workload. Therefore, we did not consider the case that the data center with the cheapest brown energy price cannot accommodate the remaining part of the requests. If this is the case, a cap/limit on the maximum rate of requests each data center can handle must be considered by the algorithm.

Algorithm 1 runs a loop on all data centers to fetch required information to calculate values of R and r_d. Then according to the condition in Line 11 makes another loop to set weight values for each data center. Therefore, the asymptotic time complexity of algorithm is $O(n)$, where n is the number of data centers.

3.2. Auto-Scaling Policy

Algorithm 2 shows the details of the auto-scaling policy used in our system. The auto-scaling policy proposed in this paper is a reactive auto-scaling working based on threshold-based rules. By conducting profiling study, we set an upper threshold below which the web server is able to provide responses within an acceptable time frame. The ratio of arrival request rate at local load balancer to the threshold value gives us the total number of required web servers.

Note that, Algorithm 2 is an algorithm without a loop or a recursion and incurs the time complexity of $O(1)$. The auto-scaler based on the profiling information sets the request rate at or below which a web server running on a specific machine type can efficiently generate responses for the web requests (Line 1). Then, in Lines 2–4, the total number of required web servers is computed according to the average request rate in the last time window (e.g., last 3 minutes). The number of required web servers is computed by diving the average arrival rate of web requests at the local load balancer to the threshold value and taking the ceiling of the result (Line 4). If total number of required web server machines is higher than the number of the currently running web servers, it allocates more web server machines (Line 7); otherwise, if the number is lower, it deallocates excess machines (Line 9). Finally, in Line 13, the local load balancer is informed by the updated list of web servers. Please note that the local load balancing between the web server machines is done in an evenly weighted round robin fashion as the server farm is homogeneous.

Algorithm 2 Auto-scaling Policy

1: $t \leftarrow$ `Set the threshold (i.e., appropriate rate for a web server) based on the profiling data`
2: $r \leftarrow$ `Fetch the request rate in the recent time window from the local HAProxy load balancer`
3: $o \leftarrow$ `Fetch the number of currently ON web server machines`
4: $m \leftarrow \lceil r \div t \rceil$ `Compute the total number of web server machines required`
5: $n \leftarrow m - o$
6: **if** $n > 0$ **then**
7: `Add` n `more web servers`
8: **else if** $n < 0$ **then**
9: `Remove` n `web servers`
10: **else**
11: `No scaling is required`
12: **end if**
13: `Inform local load balancer with the new list of web servers`

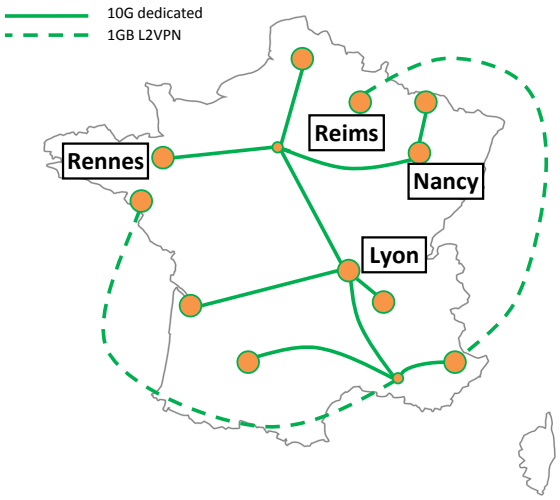


Figure 4: Grid'5000 testbed.

4. Performance Evaluation

To evaluate our system, we developed a prototype and performed experimental studies in a real testbed (Grid'5000) using real traces of requests for English Wikipedia pages. In the following, we explain details of the testbed, workload, renewable power traces, and electricity prices. Then, we present results of experiments conducted using this setup. Our aim is to understand the cost and energy consumption performance of the proposed system in realistic settings.

4.1. Experimental Setup

4.1.1. Testbed

As testbed, we used Grid'5000 [12], a French experimental grid platform.[10] Grid'5000 comprises sites geographically distributed across France. We consider a group of 3 sites equipped with power monitoring APIs [13] in the following locations: *Lyon*, *Rennes* and *Reims* shown in Figure 4. We set up the experimental testbed by preparing 3 deployable environments as follows:

[10]The Grid 5000 project, `http://www.grid5000.org/`.

Table 1: The characteristics of machines hosting web servers in different sites.

Site	CPU	Number of cores	Memory
Lyon	AMD Opteron 250 2.4GHz	2	2GB
Reims	AMD Opteron 6164 HE 1.7GHz	2	48GB
Rennes	AMD Opteron 6164 HE 1.7GHz	2	48GB

- *Database (DB)*: a mysql database server loaded with the English Wikipedia dataset as of Jan 3rd, 2008 containing roughly 2 million wiki pages and size of 3GB.
- *Web Server (WS)*: an Apache Web Server (version 2.4.10) with the installed Mediawiki application.
- Local Load Balancer (Local-LB): HAProxy (version 1.6) load balancer along with the auto-scaler Java program.

All deployable environments run on the *Debian Linux Wheezy* operating system. The characteristics of machines used in each site for hosting web servers are summarized in Table 1. To switch on/off a deployed web server, we used Grid'5000 APIs for accessing the "Wake-on-LAN" interfaces. By switching off, we set the physical machine to the "hard" power off mode (i.e., physical shut down).

To replay traces of requests by Wikipedia users, we used wikibench benchmark tool [14].[11] Wikibench is a web hosting benchmark allowing the stress-test of systems designed to host web applications. Using *wikijector* software module of wikibench, one can generate traffic by replaying traces of user requests actually made to Wikipedia. For our experimental study, we deploy wikijector on a machine in the *Nancy* site to mimic Wikipedia users sending requests to the Global-LB. Global-LB is deployed on a separate host in the *Nancy* site besides the controller Java program. The architecture of the prototype system deployed on the Grid'5000 testbed is shown in Figure 5.

4.1.2. Workload

We used real traces of requests to the web servers from the Wikimedia Foundation as workload. Our workload contains 5% of all user requests

[11] Wikibench, the realistic web hosting benchmark, `http://www.wikibench.eu/`.

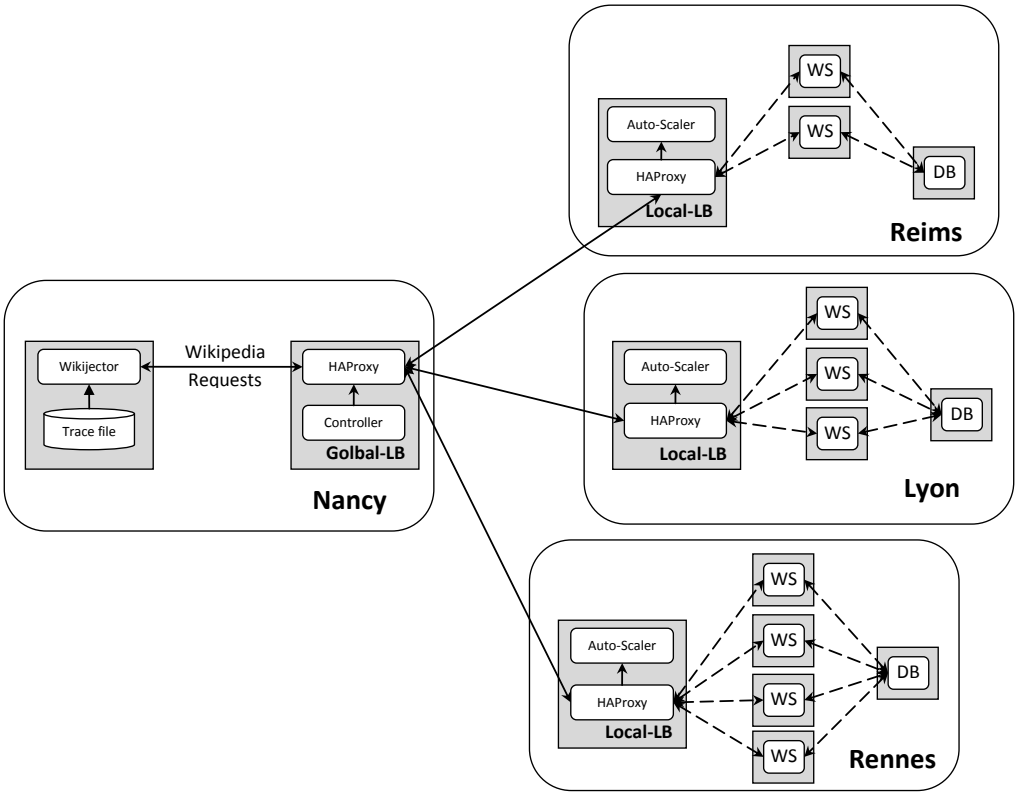


Figure 5: The architecture of the prototype system.

issued to English Wikipedia resources during the period of 19$^{\text{th}}$ to 21$^{\text{st}}$ of September 2007.[12] Figure 6 depicts the number of requests per second for the same period.

4.1.3. Availability of Renewable Energy

To capture the availability of solar energy in the location of each data center, we use data traces by SoDa Service[13] with 30 minutes granularity between 19$^{\text{th}}$ and 21$^{\text{st}}$ of Sept 2007. The Global Horizontal Irradiance (GHI)

[12] `http://www.wikibench.eu/wiki/2007-09/`

[13] http://www.soda-is.com, The SoDa Service is a broker to a list of services and web-services related to Solar Radiation proposed by several providers in Europe and abroad. The SoDa Service is provided by two mirror sites: one hosted in MINES ParisTech, Sophia Antipolis France, and the other by Transvalor S.A., Mougins.

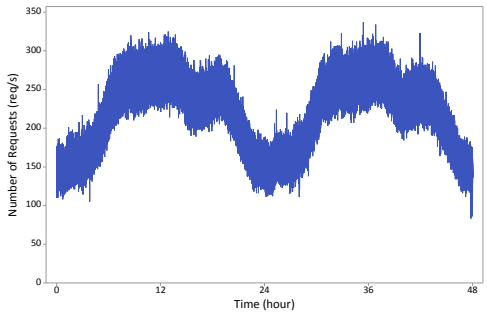
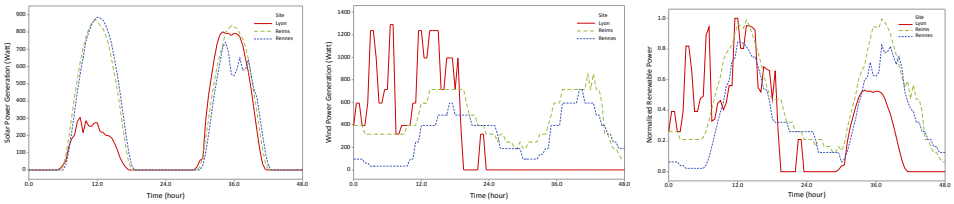


Figure 6: The English Wikipedia workload for 19^{th} and 20^{th} of September 2007.



(a) Solar (b) Wind (c) Normalized total

Figure 7: Renewable Power Generation for tow days.

in the location of each data center is used to calculate the output for solar photovoltaics (PV) power. We assume that each data center uses power generated by the PV panels of total $4000m^2$ area with tilt angle of 45° degree and PV cell efficiency of 30% (Roughly the highest efficiency reported so far [15]). We calculate the PV power module output on the tilt surface based on the model in [16].

We use meteorological data collected from Weather Underground[14] traces to model wind power for the same dates. We presume each data center uses a GE 1.5MW wind turbine to generate wind power. To estimate the average wind power production, the model proposed by Fripp and Wiser [17] is employed where the wind speed, the air temperature, and the air pressure measurements in the location of each site are fed into the model.

The summation of power generated from these two sources for every 30 minutes is computed as available renewable power for every site. Figures 7 (a), (b) and (c) show respectively the solar, wind, and normalized total power

[14] www.wunderground.com/history/

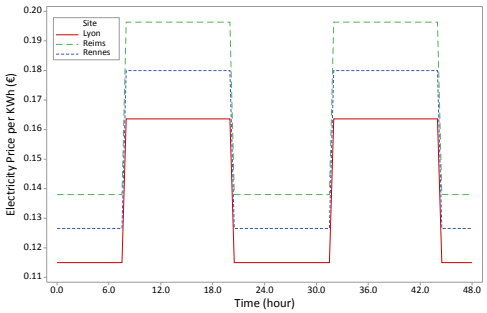


Figure 8: Electricity prices for two days.

generated from renewable sources on each site for the 2-day period. In our investigation, we scale the normalized value of renewable power availability linearly in a way that the average renewable power generation is roughly enough to serve the average workload. Accordingly, every normalized power value is multiplied by 2128.5 computed in this way.

4.1.4. Electricity Prices

In France, the utility grid power is mostly generated by Électricité de France (EDF), French utility company, and is primarily produced from nuclear power sources. Moreover, EDF offers wide range of tariffs which are consistent all over France. In our experiment, we assume on-peak/off-peak scheme, the most common type of variable energy pricing in the market. The electricity prices charged by EDF for on-peak (between 8am and 8pm) and off-peak are set to €0.1636/kWh and €0.1150/kWh, respectively. In order to incorporate price variability and effect of brown energy into our experiment, we increased the electricity price for each site by a factor based on the closeness of the site to non-renewable plants. Using geographical location of EDF non-renewable plants in France [18], we increased the electricity prices for Reims, Rennes, and Lyon, respectively, by 20%,10%, and 0%, as it is shown in the Figure 8.

4.2. Benchmark Algorithms

To evaluate the performance of the proposed GreenLB policy, we consider two benchmark algorithms.

4.2.1. Round Robin (RR)

The RR policy sets equal weights for Global-LB which results in the even distribution of load among all sites. Since Wikipedia workload and the

renewable power generated on every site exhibit a similar diurnal pattern, RR policy that evenly distributes requests among data centers is a competent benchmark policy to evaluate the performance of GreenLB.

4.2.2. Capping

Le et al. [19] proposed a policy for request distribution across data centers to minimize the overall energy cost. Similar to our method, their optimization method seeks to define the fraction of requests that should be forwarded to different data centers to minimize cost for an Internet service provider. Therefore, we decided to compare the performance of our proposed GreenLB policy with their method which we refer to as "Capping" policy from now on.

Since, their method is different from ours in some aspects and they consider brown energy caps for each data center (not considered in this paper), we modified their method in several ways to adapt it to our settings. First, we set *infinite* brown energy caps for data centers to let their policy solve the optimization problem in the absence of brown energy caps. We also considered zero cost for using green energy because in our settings data centers use free of charge on-site renewable power. Finally, the constraints related to fraction of requests are updated in a way that the minimum percentage of requests forwarded to each center is enough to utilize available renewable power on that site. If there is abundant renewable power (i.e., the renewable power generated by all data centers collectively is more than the power required to handle the entire workload), similar to our policy, we break down the load proportionally to the available renewable power for each data center as no feasible solution can be constructed by their optimization technique. According to the above modifications and assuming that the mixture of green and brown energy for the next time slot is known by the load balancer, their optimization method converts into an LP (Linear problem) which can be solved with an LP solver. Following their method, a solution for the optimization problem is periodically computed on a regular basis of once per hour. Their Auto-Regressive Integrated Moving Average (ARIMA) modeling is also used to predict the request rate for the next hour. Figure 9 shows the actual and ARIMA-predicted request rates for the two-day Wikipedia workload used in our experiments.

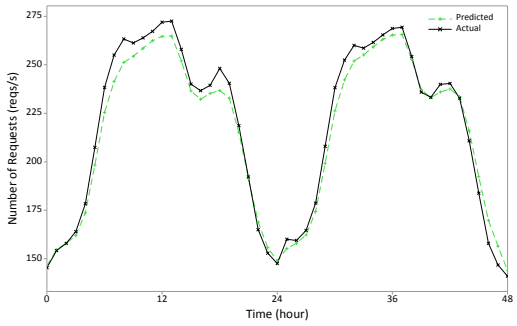


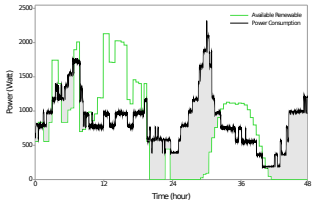
Figure 9: Hourly actual and predicated request rate used by the Capping policy.

4.3. Experimental Results and Analysis

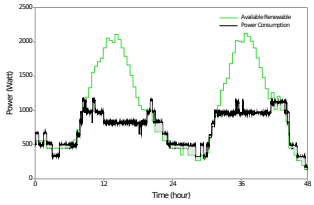
This section presents our experimental results. We run two-day experiments on the pre-configured testbed using the explained traces of the workload, renewable power and electricity prices for GreenLB, Capping, and RR policies. The auto-scaling algorithm on Local-LB are triggered every 2 minutes for all policies. The value is set based on a preliminary pilot study conducted before the main experiments. The load balancing algorithm on Global-LB is executed every 3 minutes for the proposed GreebLB policy and every 60 minutes for the Capping policy. We measure the power consumption via the servers' built-in power monitoring APIs with a granularity of 30 seconds. Please note that we only collect and report the power consumption for the web servers, since the power consumed by the load balancer and database server machines on each site are largely constant during the experiments.

Figure 10 shows the power consumption and the brown energy usage for each site when the GreenLB algorithm is used to distribute workload among sites. The GreenLB algorithm successfully follows the available renewable energy on each site as it can be clearly seen in Reims and Rennes data centers. The pattern is different for Lyon because whenever there is not enough renewable energy available, the algorithm redirects more requests to the Lyon data center which has the cheapest price of electricity per request.

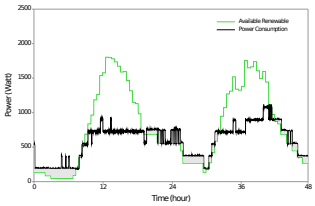
Similarly, Figures 11 and 12 illustrate that the green and brown power consumptions when Capping and RR algorithms are employed respectively. As shown in Figure 11, Capping policy demonstrates similar behavior to GreenLB, even though future knowledge regarding requests rate and availability of renewable energy for the next hour are available to this policy. This can be explained by the fact that the Wikipedia workload consists of



(a) Lyon

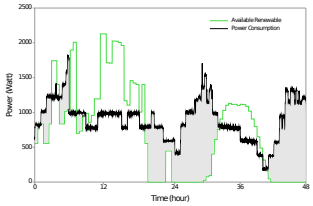


(b) Reims

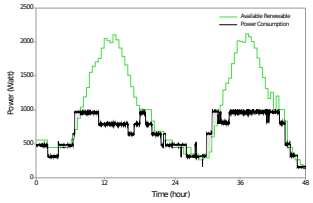


(c) Rennes

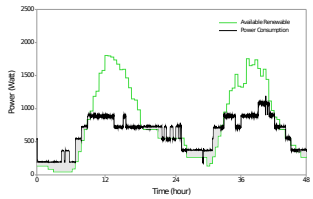
Figure 10: The power consumption for different sites using Green Load Balancing (GreenLB) algorithm. The shaded area shows brown energy usage.



(a) Lyon

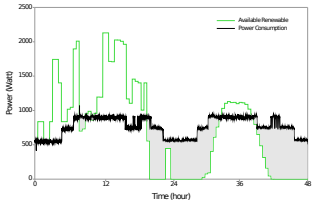


(b) Reims

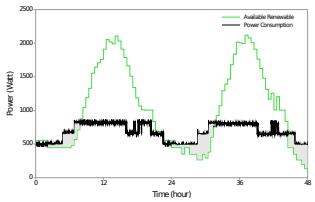


(c) Rennes

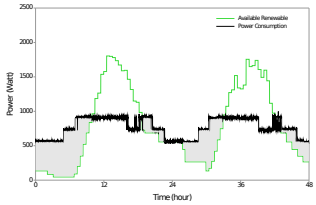
Figure 11: The power consumption for different sites using Capping algorithm. The shaded area shows brown energy usage.



(a) Lyon

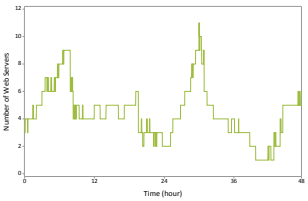


(b) Reims

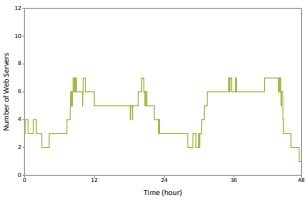


(c) Rennes

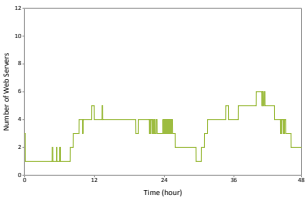
Figure 12: The power consumption for different sites using Round Robin (RR) algorithm. The shaded area shows brown energy usage.



(a) Lyon

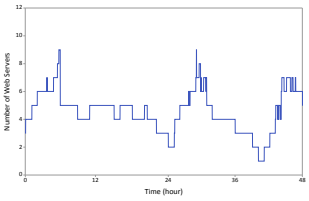


(b) Reims

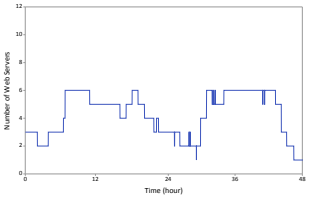


(c) Rennes

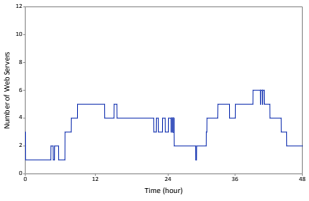
Figure 13: The number of Web Servers for different sites using Green Load Balancing algorithm.



(a) Lyon

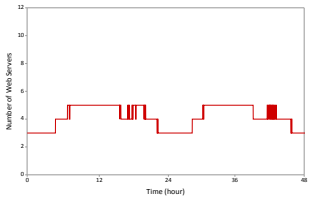


(b) Reims

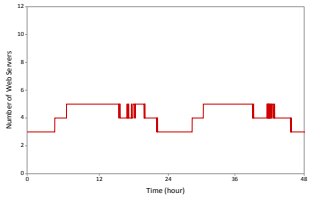


(c) Rennes

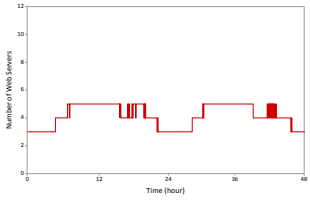
Figure 14: The number of Web Servers for different sites using Capping algorithm.



(a) Lyon



(b) Reims



(c) Rennes

Figure 15: The number of Web Servers for different sites using Round Robin (RR) algorithm.

many small sized requests that can be reactively routed among multiple data centers and in-advance proactive decision making is deemed unnecessary.

A comparison between Figures 10 and 12 clearly illustrates that substantially more brown energy usage happens in the Reims and Rennes data centers when the RR algorithm is used while less brown energy usage occurs in the Lyon data center. Figure 13,14, and 15 show the number of ON web servers for sites for GreenLB, Capping and RR policies, respectively.

Table 2 summarizes the results shown in the figures. The aggregated total and brown power consumption for all sites shows that even though all algorithms cause similar power consumptions, GreenLB uses 17% less brown energy and saves cost by almost 22% in comparison to RR. The results demonstrate that GreenLB can significantly increase green energy utilization and decrease electricity cost by 8% and 22% compared to Capping and RR policies, respectively, even when sites have similar amount of renewable power production and very competitive price of electricity. The total amount of brown power consumed by GreenLB policy is 2.63 kWh, which is 7% and 17% less than Capping and RR policies, respectively. It is expected that the difference in renewable power utilization and cost saving increases substantially whenever there exists more gaps between the renewable power availability and electricity prices of different sites.

Capping and GreenLB share major similar characteristics and provide competitive solutions for the geographical load balancing problem tackled in this paper. However, the Capping policy works based on linear optimization which is considerably less efficient than GreenLB in terms of time and space complexity. Moreover, errors imposed by future load predication and hourly based decisions performed by Capping policy result in 7% more cost and 8% more brown power usage compared to our proposed GreenLB algorithm.

In order to study the impact of our proposed load balancing algorithm on the response time, we measured the real-time response time for all the Wikipedia requests submitted to the system. Figure 16 shows the CDF of the response time observed by all load balancing algorithms. The graph demonstrates that there is no significant difference in the response time of the algorithms and majority of requests are responded within the acceptable range of hundreds of milliseconds, while RR shows marginally more stable response time. More than 90% of request are responded in about 350ms and less for the GreenLB algorithm. There are few peaks of high response time upto few seconds for all policies happening by reasons such as Java garbage collection for Wikijector module and PHP garbage collections in Apache web

Table 2: Summary of Results.

Site	Metric	RR	Capping	GreenLB
Lyon	Power Consumption (kWh)	36.2	42.9	41.2
	Brown Consumption (kWh)	13.3	19.0	16.9
	Cost (€)	1.71	2.31	2.01
Reims	Power Consumption (kWh)	32.5	32.5	35.4
	Brown Consumption (kWh)	3.1	1.1	1.9
	Cost (€)	0.42	0.15	0.27
Rennes	Power Consumption (kWh)	36.4	29.7	28.3
	Brown Consumption (kWh)	9.3	2.9	2.6
	Cost(€)	1.23	0.39	0.35
Total	Power Consumption (kWh)	**105**	**105**	**105**
	Brown Consumption (kWh)	**25.7**	**23.0**	**21.4**
	Cost(€)	**3.36**	**2.85**	**2.63**

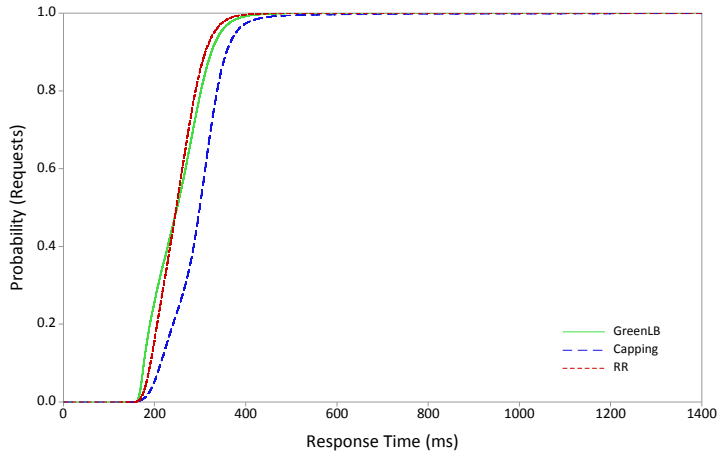


Figure 16: CDF of average response time per second for English Wikipeida requests using different algorithms.

servers.

The comprehensive analysis of the experimental results also demonstrates that the impact of the delay associated with switching on and off web servers on the response time of requests is negligible. The main reason that switching on delay is not an issue is that we chose the threshold for scaling out web servers in the auto-scaler algorithm sufficiently below the maximum capacity of the each web server. Therefore, each web server machine has some spare capacity to handle portion of extra load before it becomes fully saturated and affects the response time. This way, our auto-scaler algorithm includes an acceptable level of over-provisioning to avoid Quality of Service degradation due to switching on (booting) time of web server machines. That is, whenever the incoming request rate at the local load balancer increases, the auto-scaler (at least in case of Wikipedia workload) has enough time to scale out and to add new fully operational web servers. Switching off web server machines also does not affect the response time, as in the first step of the scale-in process, the auto scaler removes the target machine from the list of available machines in the local load balancer. Therefore, this machine will not receive additional requests. Then a signal for switching off is dispatched and the machine only switches off after all remaining requests in the web server queue are responded.

5. Related Work

Over the last decade, power management techniques to minimize data centers' costs and environmental impacts have gained considerable attention by both academia and industry. Large data centers such as those used by big companies like Google and Amazon can host thousands of physical servers and require up to tens of megawatts to power them [20]. As result, service providers are under huge pressure to reduce their energy consumption and its associated costs. This has pushed them towards using more sustainable and green data centers. In a recent study, Shuja et al [21] have provided a survey of enabling techniques and technologies for sustainable and green data centers.

Most of the early research studies on energy efficiency of data centers focus on making green data centers using optimization techniques within a single data center; techniques such as CPU dynamic voltage and frequency scaling (DVFS) [22], virtualization and VM consolidation [23, 24], and workload scheduling [25]. An extensive survey and taxonomy of these can be found in [26]. Similarly, Shuja et al. [27] conducted a survey of techniques and architectures for designing an energy-efficient data center.

5.1. Leveraging green energy

There is a number of studies focused on reducing brown energy or power consumption, monetary costs, and environmental impact using renewable sources of energy for different types of applications, for example, batch processing [28, 29, 30], and interactive processing [31]. Goiri et al. [28] present Parasol, a green data center prototype. They define a scheduler for planning the workload execution and for selecting the energy sources: solar panels, batteries or grid. The scheduler makes decisions based on workload and energy predictions, battery level, DC characteristics and grid electricity prices. Liu et al. [29] present an approach to model the energy flows in a data center in order to optimize its operation. They predict renewable energy and IT demand to schedule IT workload and allocates IT resources within a data center according to time varying power supply and cooling efficiency. Authors in [30] propose GreenSlot a parallel batch job scheduler for a data center powered by a photovoltaic solar array and the utility grid. Stewart and Shen [31] also try to maximize green energy use in data center for interactive Internet services. All these studies focus on load or demand shifting

to maximize green energy utilization within one data center, while we focus on multiple data center load redirection.

5.2. Geographical load balancing (GLB)

A large body of recent literature focuses on reducing energy costs targeting geographically distributed data centers. This group of work mainly devises techniques for workload distribution across geo-distributed data centers in order to achieve performance objectives such as minimizing cost, maximizing renewable energy utilization, and minimizing emission. Rahman et al. [32] present a comprehensive survey on data center power management using geographic load balancing.

Among different approaches of geographical load balancing, "following the renewables" has gained considerable attentions. This approach requires that dynamic load balancing mechanism be aware of the availability of renewable energy at data centers [21]. One of the early studies on GLB is done by Liu et al. [4]. Using GLB, they propose algorithms to maximize renewable energy utilization and show how dynamic electricity price can affect brown energy usage. They use trace-based numerical simulations to evaluate their algorithms. An extension to this work has been done by Lin et al. [3], where they propose online algorithms to exploit the potential of geographical diversity of internet-scale services on renewable energy utilization. As part of their research, they show the optimal portfolio of solar and wind energy sources in GLB. Similarly, He et al. [33] considered the sustainability of data centers by proposing socially-responsible load scheduling for data centers where they consider emission cost as the social cost. Chen et al. [34] proposed a scheduling algorithm that considers the workload fluctuation, jobs' deadline, variable green energy supply, outside temperature, and data center cooling dynamics. In a recent comprehensive study, Paul et al. [35] proposed a holistic framework for dynamic load distribution using online algorithms techniques. To minimize cost, they exploit the spatial variation in electricity price and renewable energy generation for a cloud service provider having a large number of data centers collocated with renewable energy sources. Their approach not only maximizes green energy utilization, but also minimizes the number of server switching. They have conducted extensive simulations with real data traces to evaluate their system. Berral et al. [36] go one step further by proposing a framework that offers suitable locations of data centers to a provider seeking to create a network of sustainable data centers for a follow-the-renewables HPC cloud service.

All these studies consider data centers with on-site free of charge power generations from renewable energy sources. Similar to these, we consider GLB to reduce energy cost and to maximize renewable energy utilization. However, we mainly focus on practical considerations and we evaluate our proposed system in a real environment using realistic traces of workload, renewable energy and electricity prices.

5.3. Power capping

Another set of research efforts on dynamic load balancing assume that data center must pay for the power drawn from the off-site renewable energy sources. Le et al. [37] considered load distribution across data center sites by including limiting the energy usage from non-renewable sources. Gao et al. [38] propose a framework, which is compared to the method by Le et al. [37], for request-routing and traffic engineering considering changes in workload and carbon footprint. They attempt to balance the three-way trade-off between access latency, carbon footprint, and electricity costs. A recent topic of interest that explores capping the brown energy consumption has been studied by several studies such as [19, 39]. Policies used in this category propose techniques to abide by carbon caps on brown energy consumption. Le et al. [19] propose a software framework to distribute requests among different data centers while supporting the capping brown energy consumption. The goal is to abide by the carbon caps for each data center without excessively increasing costs or degrading performance. Similarly, Abbasi et al. [39], propose online algorithms to tackle the same problem. They remove the need for long-term future prediction via exploiting their proposed online algorithms. All these research efforts assume that data centers have to pay for renewable energy from off-site utility power. Moreover, electricity prices for power generated from off-site renewable sources might be higher than brown power. However, we assume data centers are equipped with on-site renewable power facilities and power generated in this way is free of charge.

5.4. Online application load balancing

Efficient Use of geographically spread resources for online interactive application such as web applications has been explored by several authors. Goudarzi and Pedram [40] investigate the load balancing problem for online service applications using a cloud system comprised of geographically dispersed data centers. Offline and online algorithms are proposed to determine

the application placement and migration based on renewable energy generation capacities at different data centers in the cloud system. Similar to our work, Kanizo et al. [41] use Wikimedia access logs to test their threshold-based load sharing techniques. However, they did not consider renewable energy power generation in their work. Using the same workload, Zhang et al. [42] proposed GreenWare, a framework to maximize renewable energy utilization of Geo-distributed data centers. Contrary to our work, they use simulation for evaluation purposes and assume that renewable energy is more expensive than brown energy.

6. Conclusions and Future Work

We proposed a cost and energy efficient load balancing algorithm to distribute web applications requests among multiple data centers geographically distributed in a region. A prototype was implemented and experimental studies in a real testbed are performed using it. A group of 3 data centers from Grid5000 testbed scattered in France and equipped with power monitoring APIs were selected. Meteorological data in the location of each data center used to model solar and wind power generation. Real traces of web requests for English Wikipedia pages were replayed to generate the workload. Our proposed green load balancing algorithm is employed to distribute load among data centers based on the availability of renewable energy and prices of grid brown energy for each site. Finally, total power consumption, brown power usage, and cost of electricity are measured and compared to two benchmark algorithms. Results showed that, even in case similar amount of renewable power production in sites and very competitive price of electricity, our proposed policy is able to reduce the cost by 22% and 8% and brown energy by 17% and 7% in comparison to round robin and policy proposed in [19], respectively. We also demonstrated that the average response time of requests is not affected by our proposed load balancing algorithm.

Studies like the one presented in this article can be used to measure the required amount of energy to handle certain workload. Accordingly, this provides insights regarding the renewable energy facilities required by each data center to handle the load. Future research needs to be done to incorporate our proposed method with support for investment decisions on establishing renewable power generators in a real-world system.

In our proposed system, we only considered stateless web applications in which requests can be responded individually regardless of which user is-

sued them. In future, we focus on the design and implementation of "Sticky load balancing" policies in which after a session between a client and an application server is established, all subsequent requests from this session are redirected to the same server. Moreover, in this paper, we limit our geographical load balancing to data centers dispersed throughout a small region (e.g., France or Europe) where routing requests among different data centers will not significantly affect response time as network delays remains in an acceptable range. We are interested to extend our system for worldwide geographical load balancing in which network proximity of the user is considered in the policy.

The rest of our future work will focus on renewable energy-aware geographical load balancing for other types of workloads/applications such as bag of tasks, scientific workflows, where platforms and tools such as Aneka [43] or workflow engine [44] can be employed. Demand response and capping the brown power consumption to reduce carbon footprint and to promote carbon neutrality are among recent topics of interests which can also be considered as future research targets.

Acknowledgments

This work was partially supported by Australian Research Council (ARC) Future Fellowship and Discovery Project grants. Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by INRIA and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr). Authors would like to thank Adam Wierman from Caltech for his inspiring thoughts on this work and David Margery and Laurent Lefevre from Inria for their technical support of the experiments. They would also like to thank Rodrigo N. Calheiros, the editors of this journal, and three anonymous reviewers for their many helpful comments and suggestions on an earlier version of this article.

References

[1] NRDC, Anthesis, Scaling up energy efficiency across the data center industry: Evaluating key drivers and barriers, Tech. rep., Natural Resources Defense Council (2014).

[2] I. Goiri, K. Le, T. D. Nguyen, J. Guitart, J. Torres, R. Bianchini, Greenhadoop: Leveraging green energy in data-processing frameworks, in: Proceedings of the 7th ACM European Conference on Computer Systems, EuroSys '12, ACM, 2012, pp. 57–70. doi:10.1145/2168836.2168843.

[3] M. Lin, Z. Liu, A. Wierman, L. Andrew, Online algorithms for geographical load balancing, in: Proceedings of the International Green Computing Conference, IGCC '12, 2012, pp. 1–10. doi:10.1109/IGCC.2012.6322266.

[4] Z. Liu, M. Lin, A. Wierman, S. H. Low, L. L. Andrew, Greening geographical load balancing, in: Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '11, ACM, 2011, pp. 233–244. doi:10.1145/1993744.1993767.

[5] M. Adnan, R. Sugihara, R. Gupta, Energy efficient geographical load balancing via dynamic deferral of workload, in: Proceedings of the 5th IEEE International Conference on Cloud Computing, CLOUD '12, 2012, pp. 188–195. doi:10.1109/CLOUD.2012.45.

[6] M. Lin, A. Wierman, L. L. H. Andrew, E. Thereska, Dynamic right-sizing for power-proportional data centers, in: Proceedings of INFOCOM 2011, 2011, pp. 1098–1106. doi:10.1109/INFCOM.2011.5934885.

[7] Z. Liu, M. Lin, A. Wierman, S. H. Low, L. L. Andrew, Geographical load balancing with renewables, SIGMETRICS Perform. Eval. Rev. 39 (3) (2011) 62–66. doi:10.1145/2160803.2160862.

[8] N. Grozev, R. Buyya, Multi-cloud provisioning and load distribution for three-tier applications, ACM Transactions on Autonomous and Adaptive Systems (TAAS) 9 (3) (2014) 13:1–13:21. doi:10.1145/2662112.

[9] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, T. Wood, Agile dynamic provisioning of multi-tier internet applications, ACM Transactions on Autonomous and Adaptive Systems (TAAS) 3 (1) (2008) 1:1–1:39. doi:10.1145/1342171.1342172.

[10] J. Jiang, J. Lu, G. Zhang, G. Long, Optimal cloud resource auto-scaling for web applications, in: Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid '13, 2013, pp. 58–65. doi:10.1109/CCGrid.2013.73.

[11] H. Fernandez, G. Pierre, T. Kielmann, Autoscaling web applications in heterogeneous cloud infrastructures, in: Proceedings of IEEE International Conference on Cloud Engineering, IC2E '14, 2014, pp. 195–204. doi:10.1109/IC2E.2014.25.

[12] R. Bolze, F. Cappello, E. Caron, M. Daydé, F. Desprez, E. Jeannot, Y. Jégou, S. Lanteri, J. Leduc, N. Melab, et al., Grid'5000: a large scale and highly reconfigurable experimental grid testbed, International Journal of High Performance Computing Applications 20 (4) (2006) 481–494. doi:10.1177/1094342006070078.

[13] F. Clouet, S. Delamare, J.-P. Gelas, L. Lefevre, L. Nussbaum, C. Parisot, L. Pouilloux, F. Rossigneux, A unified monitoring framework for energy consumption and network traffic, in: International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities, TridentCom '15, 2015, p. 10.

[14] E.-J. van Baaren, Wikibench: A distributed, wikipedia based web application benchmark, Master's thesis, VU University Amsterdam (2009).

[15] M. A. Green, K. Emery, Y. Hishikawa, W. Warta, E. D. Dunlop, Solar cell efficiency tables (version 45), Progress in photovoltaics: research and applications 23 (1) (2015) 1–9. doi:10.1002/pip.2573.

[16] Solar Radiation on a Tilted Surface, `http://pveducation.org/pvcdrom/properties-of-sunlight/solar-radiation-on-tilted-surface`.

[17] M. Fripp, R. H. Wiser, Effects of temporal wind patterns on the value of wind-generated electricity in california and the northwest, IEEE Transactions on Power Systems 23 (2) (2008) 477–485. doi:10.1109/TPWRS.2008.919427.

[18] EDF, Fossil-fired energy.
URL `http://energie.edf.com/fichiers/fckeditor/Commun/En_`

Direct_Centrales/collection_nos_energies/edf_thermique_bd_va.pdf

[19] K. Le, R. Bianchini, T. D. Nguyen, O. Bilgir, M. Martonosi, Capping the brown energy consumption of internet services at low cost, in: Proceedings of the International Green Computing Conference, IGCC '10, 2010, pp. 3–14. doi:10.1109/GREENCOMP.2010.5598305.

[20] F. Kong, X. Liu, A survey on green-energy-aware power management for datacenters, ACM Computing Surveys 47 (2) (2014) 30:1–30:38. doi:10.1145/2642708.

[21] J. Shuja, A. Gani, S. Shamshirband, R. W. Ahmad, K. Bilal, Sustainable cloud data centers: A survey of enabling techniques and technologies, Renewable and Sustainable Energy Reviews 62 (2016) 195–214. doi:10.1016/j.rser.2016.04.034.

[22] C.-M. Wu, R.-S. Chang, H.-Y. Chan, A green energy-efficient scheduling algorithm using the dvfs technique for cloud datacenters, Future Generation Computer Systems 37 (2014) 141–147. doi:10.1016/j.future.2013.06.009.

[23] A. Beloglazov, R. Buyya, Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints, IEEE Transactions on Parallel and Distributed Systems 24 (7) (2013) 1366–1379. doi:10.1109/TPDS.2012.240.

[24] S. Srikantaiah, A. Kansal, F. Zhao, Energy aware consolidation for cloud computing, in: Proceedings of the USENIX 2008 conference on Power aware computing and systems, Vol. 10 of HotPower '08, San Diego, California, 2008.

[25] M. Ghamkhari, H. Mohsenian-Rad, Energy and performance management of green data centers: A profit maximization approach, IEEE Transactions on Smart Grid 4 (2) (2013) 1017–1025. doi:10.1109/TSG.2013.2237929.

[26] A. Beloglazov, R. Buyya, Y. C. Lee, A. Zomaya, A taxonomy and survey of energy-efficient data centers and cloud computing systems, Advances in computers 82 (2) (2011) 47–111.

[27] J. Shuja, K. Bilal, S. A. Madani, M. Othman, R. Ranjan, P. Balaji, S. U. Khan, Survey of techniques and architectures for designing energy-efficient data centers, IEEE Systems Journal 10 (2) (2016) 507–519. doi:10.1109/JSYST.2014.2315823.

[28] I. Goiri, W. Katsak, K. Le, T. D. Nguyen, R. Bianchini, Parasol and greenswitch: Managing datacenters powered by renewable energy, SIGARCH Comput. Archit. News 41 (1) (2013) 51–64. doi:10.1145/2490301.2451123.

[29] Z. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, Z. Wang, M. Marwah, C. Hyser, Renewable and cooling aware workload management for sustainable data centers, in: Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '12, ACM, 2012, pp. 175–186. doi:10.1145/2254756.2254779.

[30] I. Goiri, K. Le, M. E. Haque, R. Beauchea, T. D. Nguyen, J. Guitart, J. Torres, R. Bianchini, Greenslot: Scheduling energy consumption in green datacenters, in: Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, SC '11, ACM, 2011, pp. 20:1–20:11. doi:10.1145/2063384.2063411.

[31] C. Stewart, K. Shen, Some joules are more precious than others: Managing renewable energy in the datacenter, in: Proceedings of the Workshop on Power Aware Computing and Systems (HotPower) in 22nd ACM Symposium on Operating Systems Principles, SOSP '09, 2009, pp. 15–19.

[32] A. Rahman, X. Liu, F. Kong, A survey on geographic load balancing based data center power management in the smart grid environment, IEEE Communications Surveys Tutorials 16 (1) (2014) 214–233. doi:10.1109/SURV.2013.070813.00183.

[33] J. He, X. Deng, D. Wu, Y. Wen, D. Wu, Socially-responsible load scheduling algorithms for sustainable data centers over smart grid, in: Proceedings of the Third IEEE International Conference on Smart Grid Communications, SmartGridComm '12, 2012, pp. 406–411. doi:10.1109/SmartGridComm.2012.6486018.

[34] C. Chen, B. He, X. Tang, Green-aware workload scheduling in geographically distributed data centers, in: Proceedings of 4th IEEE International Conference on Cloud Computing Technology and Science, CloudCom '12, 2012, pp. 82–89. doi:10.1109/CloudCom.2012.6427545.

[35] D. Paul, W.-D. Zhong, S. K. Bose, Energy efficiency aware load distribution and electricity cost volatility control for cloud service providers, Journal of Network and Computer Applications 59 (2016) 185–197. doi:10.1016/j.jnca.2015.08.012.

[36] J. L. Berral, . Goiri, T. D. Nguyen, R. Gavald, J. Torres, R. Bianchini, Building green cloud services at low cost, in: Proceedings of the 34th IEEE International Conference onDistributed Computing Systems, ICDCS '14, 2014, pp. 449–460. doi:10.1109/ICDCS.2014.53.

[37] K. Le, O. Bilgir, R. Bianchini, M. Martonosi, T. D. Nguyen, Managing the cost, energy consumption, and carbon footprint of internet services, in: Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '10, ACM, 2010, pp. 357–358. doi:10.1145/1811039.1811085.

[38] P. X. Gao, A. R. Curtis, B. Wong, S. Keshav, It's not easy being green, SIGCOMM Computer Communication Review 42 (4) (2012) 211–222. doi:10.1145/2377677.2377719.

[39] Z. Abbasi, M. Pore, S. K. S. Gupta, Online server and workload management for joint optimization of electricity cost and carbon footprint across data centers, in: Proceedings of the 28th IEEE International Parallel and Distributed Processing Symposium, IPDPS '14, 2014, pp. 317–326. doi:10.1109/IPDPS.2014.42.

[40] H. Goudarzi, M. Pedram, Geographical load balancing for online service applications in distributed datacenters, in: Proceedings of the 6th IEEE International Conference on Cloud Computing, Cloud '13, 2013, pp. 351–358. doi:10.1109/CLOUD.2013.77.

[41] Y. Kanizo, D. Raz, A. Zlotnik, Efficient use of geographically spread cloud resources, in: Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid '13, 2013, pp. 450–457. doi:10.1109/CCGrid.2013.18.

[42] Y. Zhang, Y. Wang, X. Wang, Greenware: Greening cloud-scale data centers to maximize the use of renewable energy, in: F. Kon, A.-M. Kermarrec (Eds.), Middleware 2011, Vol. 7049 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 143–164. doi:10.1007/978-3-642-25821-3_8.

[43] R. Buyya, D. Barreto, Multi-cloud resource provisioning with aneka: A unified and integrated utilisation of microsoft azure and amazon EC2 instances, in: Proceedings of the 2015 International Conference on Computing and Network Communications, CoCoNet '15, IEEE, 2015, pp. 216–229. doi:10.1109/CoCoNet.2015.7411190.

[44] S. Pandey, D. Karunamoorthy, R. Buyya, Workflow engine for clouds, in: R. Buyya, J. Broberg, A. Goscinski (Eds.), Cloud Computing: Principles and Paradigms, John Wiley & Sons, Inc., 2011, pp. 321–344. doi:10.1002/9780470940105.ch12.