Exploiting Green Energy to Reduce the Operational Costs of Multi-Center Web Search Engines

Roi Blanco
Yahoo Labs
London, UK
roi@yahoo-inc.com

Matteo Catena
Gran Sasso Science Institute
L'Aquila, Italy
matteo.catena@gssi.infn.it

Nicola Tonellotto
ISTI-CNR
Pisa, Italy
nicola.tonellotto@isti.cnr.it

ABSTRACT

Carbon dioxide emissions resulting from fossil fuels (brown energy) combustion are the main cause of global warming due to the greenhouse effect. Large IT companies have recently increased their efforts in reducing the carbon dioxide footprint originated from their data center electricity consumption. On one hand, better infrastructure and modern hardware allow for a more efficient usage of electric resources. On the other hand, data-centers can be powered by renewable sources (green energy) that are both environmental friendly and economically convenient.

In this paper, we tackle the problem of targeting the usage of green energy to minimize the expenditure of running multi-center Web search engines, i.e., systems composed by multiple, geographically remote, computing facilities.

We propose a mathematical model to minimize the operational costs of multi-center Web search engines by exploiting renewable energies whenever available at different locations. Using this model, we design an algorithm which decides what fraction of the incoming query load arriving into one processing facility must be forwarded to be processed at different sites to use green energy sources.

We experiment using real traffic from a large search engine and we compare our model against state of the art baselines for query forwarding. Our experimental results show that the proposed solution maintains an high query throughput, while reducing by up to ∼25% the energy operational costs of multi-center search engines. Additionally, our algorithm can reduce the brown energy consumption by almost 6% when energy-proportional servers are employed.

Keywords

Web search engines, query forwarding, green energy

General Terms

Search engine architectures and scalability

WWW 2016, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4143-1/16/04.
http://dx.doi.org/10.1145/2872427.2883021.

1. INTRODUCTION

Large commercial Web search engines receive a vast amount of queries every day which are executed against an ever growing crawled and indexed collection of Web pages. One main challenge of modern search engine architectures is to deliver results back to users maintaining responses times that are within a fraction of a second.

In order to be able to meet user latency expectations [26] search engines comprise thousands of servers which in turn are organized within several clusters and process queries in a distributed manner [3]. These search clusters are hosted in large data centers alongside with their associated infrastructures such as telecommunications, power supplying, thermal cooling and fire suppression among others [2]. Query response times can be further reduced if large-scale Web search engines distribute their infrastructures and operations across several, geographically distant, data centers. Each site maintains an index replica of the most recent crawling of the Web [5] and user queries are subsequently processed in the closest available data center to yield low query response latencies.

While distributed facilities are the key to enable large-scale search they raise economical issues in the form of maintenance, depreciation and operational costs. As a matter of fact, the electricity expenditure to power a data center can exceed its original investment cost [21], accounting for over 10% of the total ownership cost [16]. Therefore, energy prices may play a role in deciding where to build a data center, favoring countries where energy is cheaper. In fact, energy shows both spatial and temporal price variations, with its cost changing during the day due to supply/demand factors [24]. *Query forwarding* has been proposed as a factor to reduce the operational costs of search engines [15, 29]. The main idea is to dispatch queries from the data center that firstly received the request to a different one. In order to reduce the energy expenditure, query forwarding aims to shift the query load towards those data centers which incur in the lowest energy price at every time step.

However, operational costs are not the only concern for data center owners as the power consumption of these facilities also raise environmental concerns. For instance, the ICT sector has been reported to be responsible for roughly 2% of global carbon emissions in 2007, with general purpose data centers accounting for 14% of the ICT footprint [11]. Unsurprisingly, governments have drawn codes of conduct and best practices for those large computing facilities [10, 31] since carbon dioxide emissions are the main cause of global warming due to the greenhouse effect.

Recently, Web companies have adopted eco-friendly policies to reduce data centers energy consumption and the relative carbon footprint. These range from designing more energy efficient data centers to increasing the usage of *green energy* – auto-produced or bought from local providers.[1]

Green energy comes from resources which are renewable and do not emit carbon dioxide, such as sunlight and wind. On the contrary, *brown energy* is produced using polluting resources like carbon or oil. Since green energy sources are freely available, and thanks to governments incentives, green energy is often cheaper than brown alternatives [13, 28]. However, in many cases green energy availability can fluctuate over the day: for instance, solar and wind energy production is susceptible to weather conditions. Given that data center energy consumption depends on its usage [2], additional – possibly brown – energy needs to be bought from the energy market, when there is not enough available green energy to sustain the data center workload.

In this work we target the energy-related operational costs of large-scale, multi-center Web search engines with replicated indexes. Our research is mainly motivated by the following observations:

- green energy is available at different sites in limited quantities;
- green energy usage is assumed to be more convenient than brown energy;
- query workloads of search data centers show spatial-temporal variations, i.e., the workload of a data center varies during the day and some data centers may be under high traffic while others are mostly idle [27].

We propose a new query forwarding algorithm that aims to reduce the overall energy operational costs of a multi-center Web search engine. Our approach exploits both the green energy sources available at different sites and the differences in market energy prices. The problem of exploiting green/brown energy to reduce costs when forwarding queries is modeled as a Minimum Cost Flow Problem, taking into account the different and limited processing capacities of data centers, query response time constrains and communication latencies among sites. We evaluate the proposed algorithms using workloads obtained from the Yahoo search engine together with realistic electric price data, and we compare it with a state-of-the-art baseline for query forwarding in distributed search engines. Our results show that, in general, query forwarding plays only a little role in reducing the carbon footprint of current data centers. In other words, we provide an additional evidence of the need for more energy-proportional hardware, i.e., for hardware which consume little or no energy when not being in use [2, 4]. More importantly, we show that our solution obtains energy expenditure reductions that range from ∼15% to ∼25% with respect to standard multi-center Web search engines, outperforming the state-of-the-art.

The rest of the paper is structured as follows: Section 2 provides background information and discusses the related work. Section 3 proposes a model for a distributed search engine, its energy consumption and its operational costs, while Section 4 exploits such models to design a query forwarding algorithm leveraging the green energy available to the search engine. Section 5 illustrate our experimental setup and Section 6 reports on the results of our comprehensive evaluation. Finally, the paper concludes in Section 7.

[1] `https://www.google.com/green/`
`http://www.microsoft.com/environment/`
`https://sustainability.fb.com`

2. BACKGROUND AND RELATED WORK

Despite the economical and environmental challenges posed by data centers, there is only a limited body of work addressing these topics from the perspective of Web search engines. Barroso et al. [3] introduced one of the first studies on the energy consumption issues incurred by commercial search engines, detailing several challenges related to high density packaging, power consumption and cooling issues. Some ballpark figures report that a standard server rack of 40 machines consume around 10 MWh of power per month. Chowdhury [9] proposed the first research agenda on Green Information Retrieval, advocating for the need of Green IT and how cloud computing can play a key role in reducing the environmental impact of search technologies. In line with such agenda, Freire et al. [12] introduced a self-adaptive model to manage the number of active query servers in a fully replicated search engine while guaranteeing acceptable response times. Their method exploits the historical and current query loads of the system to autonomously decide whether to activate a query server or put it in standby. The model is formulated as a decision problem which tries to optimize the power/latency trade off, by estimating future query arrival and service times. Overall, the approach brings an energy saving of 33% with respect to a naive baseline where query servers are always active. Similarly, Lo et al. [18] proposed a feedback-based model that trade offs power savings for longer query response time in Web search engines. Their approach dynamically scale servers' CPUs frequencies, so that latency requirements are barely met under any workload. On a large server cluster the centralized implementation of this technique helps saving 20% in power consumption, while 40% savings are expected for a distributed implementation. In the context of multi-center search engines these techniques can be employed in conjunction with query forwarding in order to deploy more power-efficient architectures.

Geographically distributed search engines have been described in previous works [1, 7, 8, 14] as an alternative solution to centralized search engine infrastructures. These works describe the architecture of a multi-site Web search engine together with different query forwarding algorithms to improve the effectiveness of the search results and/or the system performance. While tackling efficiency issues, these previous works do not explicitly address the operational costs of the search infrastructure, i.e. the ongoing costs for operating the system.

Regarding the impact of financial considerations on multi-site search engines, some recent studies have suggested how to take advantage of the fact that energy prices and query volumes varies over the day [15, 29]. Kayaaslan et al. [15] investigated the possibility of dynamically shifting the query workload between data centers in order to minimize total energy cost incurred by the search engine. In their approach every index is replicated and a data center forwards queries to remote sites with cheaper energy prices. The probability of forwarding a query towards a particular data center is proportional to the amount of queries processable by the remote site and evenly shared among sites with an higher energy prices. Since queries are forwarded using a probability distribution, we call this technique PROB in the rest of the

paper. Kayaaslan et al. show that multi-sites search engines can save from 12% to 35% in energy expenditure by using their PROB approach when compared to IR systems which always locally solve the incoming queries. However, their solution does not take into account the presence of green energy sources. Teymorian et al. proposed the Rank-Energy Selective Query forwarding (RESQ) algorithm, designed to work with partially replicated indices [29]. RESQ aims to maximize ranking quality of search results, while satisfying energy cost budget constraints. Simulations show that RESQ can achieve more than 40% energy cost savings with respect to [8]. However, Teymorian et al.'s work uses query workloads that are not realistic, since they are randomly assigned to the data centers by sampling the AOL dataset. Moreover, RESQ do not perform any query forwarding in the case of fully replicated indices, therefore providing no benefits in such scenario. Sazoglu et al. [25] took into account dynamic query arrivals and electricity costs and presented a financial cost metric to measure the price of cache misses in search engines. Interestingly, the benefits provided by cost-aware caching strategies are more evident when there is an high variation in the query costs.

In the context of generic Internet services, a larger amount of literature exists targeting the power consumption of general-purpose data centers. Due to space limitations, we refer the reader to [2, 22] for more specific references. Instead, we here discuss two works particularly related to ours [17, 24].

Qureshi et al. [24] argued that Internet services can exploit variation in energy prices for cutting down their electric bill. Additionally, Le et al. [17] described a linear programming-based solution to cap brown energy consumption, without excessively increasing costs nor degrading performance. Our work differs from these as we take into account aspects specific to Web search engines. Indeed, we explicitly consider that search engines have strict response time constraints [26] and query result quality can be degraded when these requirements cannot be met [15]. Also, we conduct our experiments using real-life query workloads and arrival patterns. Finally, we perform request forwarding using a novel approach based on a Minimum Cost Flow Problem formulation.

3. PROBLEM MODEL

We model the infrastructure of a search engine as a geographically distributed system. We assume that the underlying systems are able to communicate and exchange workload using *query forwarding*, which we aim to leverage in conjunction with different pricing of electricity and availability of green energy around the world. The main goal is to use optimally the amount of green energy available at different sites to minimize the operational costs and, as a side effect, the carbon footprint of the whole infrastructure. The model analytically captures the global state of a search engine infrastructure through a set of state variables. We will implicitly assume that our model is statically valid during a fixed time length or *time slot* Δt.

Search Engine Model. We model a geographically distributed search engine as a set of data centers $\mathcal{D} = \{D_1, \ldots, D_N\}$. These data centers are placed in different and distant locations across the planet. Each data center D_i is a pair (F_i, B_i), composed by a frontend F_i and a backend B_i (Figure 1).[2] A frontend acts as a requests router: it receives user queries and decides to whether forward them to its corresponding local backend or to some remote backend. In practice, this implies that every frontend is connected through the network to every backend. A backend is composed by several server clusters, which perform the computation required to process an incoming query. Once the results are computed, these are sent back to the frontend that received the original request, which will deliver the results back to the user who issued the query.

Figure 1: Example of a Web search engine infrastructure model.

Each frontend F_i collects queries from users geographically close to data center D_i. These users act like a query source S_i sending to F_i a certain number of search request a_i over the time slot Δt. The number of queries submitted varies throughout the day, being higher at daytime than at nighttime [27]. Due to timezone differences, in the same instant, data centers will experience different query workloads.

After receiving a query, F_i may decide to forward this request to any other backend B_j or to process it locally. Our goal is to determine x_{ij}, i.e., how many search requests from frontend F_i are routed to backend B_j.

In this formulation, we enforce that the following three balance constraints hold.

1. Forwarded search requests can not be negative nor fractional, i.e., for all i and j we must have $x_{ij} \in \mathbb{N}$.

2. The sum of search requests dispatched by frontend F_i must correspond to the search requests received from the source S_i, i.e., for all i we must have:

$$\sum_{j=1}^{N} x_{ij} = a_i \ . \tag{1}$$

3. At each time slot, the backend B_j processes y_j queries, which come from the different frontends. Therefore, for all j the following holds:

$$y_j = \sum_{i=1}^{N} x_{ij} \ . \tag{2}$$

[2]Free indexes are assumed to run from 1 to N if not specified otherwise.

Any backend B_j is composed by m_j machines dedicated to process queries. Whenever a query arrives it is processed as soon as possible by a server in first-in first-out order. If every server is busy processing other requests, the query will wait in a queue. Since users expect results to be delivered in a short amount of time and queries can not wait indefinitely to be processed [26], each request must be processed within a certain *time budget* τ since its arrival on a frontend. Consequently, a query can be classified according to the time spent in processing it, and the quality of the produced results:

- *successful query* – a server processes the query and successfully terminates within τ milliseconds, generating a complete list of results;
- *approximated query* – a server starts processing the query, but the time budget τ expired before full evaluation. The query processing is early terminated and a partial list of results is generated [15];
- *failed query* – the query spends all its time budget waiting in queue. No server is able to process the query resulting in an empty result list or an error page returned to the user.

In our approach we wish to limit the number of approximated and failed queries as they can negatively impact the user satisfaction. Arguably, the configuration of the system should keep the amount of failed queries close to zero as they might severely hurt the user perception of performance.

Once the result list is generated, the backend B_j sends it back to the frontend F_i which has forwarded the relative query. The underlying communication network introduces latencies which implies that round trip times from D_i to D_j needs to be accounted for. $RTT(i,j)$ is the overall time required by the network to send a request from F_i to B_j, and the response back from B_j to F_i. Round trip times may negatively impact query processing, as they consume a part of the time budget of each forwarded query.[3]

Finally, the number m_j of identical machines hosted by the backend B_j determines V_j, the maximum query volume the backend can sustain, this is, V_j is the maximum number of queries per second that B_j can successfully process before their time budget allowance expires:

$$V_j = \frac{m_j}{\mu} \ , \tag{3}$$

where μ is the average query processing time.

The amount of queries y_j received by backend B_j should not exceed its capacity V_j. Whenever this happens the backend becomes overloaded with requests and starts to produce approximated and failed queries. Therefore, for every j, we impose that:

$$y_j \leq V_j \ . \tag{4}$$

Moreover, we want any data center to be able to process its locally assigned queries without being overloaded with those forwarded by other frontends. Given so, a remote frontend F_i should limit the number x_{ij} of queries forwarded to B_j by a quantity bounded by B_j's residual capacity, i.e., the difference between the backend capacity and the locally incoming queries, divided by the number of remote sites. In line with [15], for any $i \neq j$ we impose that:

$$x_{ij} \leq \left\lceil \frac{\max(0, V_j - a_j)}{N-1} \right\rceil \tag{5}$$

[3] $RTT(i,i)$ is assumed to be zero.

Energy Model. In this work we consider the energy consumed by a data center to be accounted exclusively to its correspondent backend. Even if this assumption does not hold in practice (for example, a considerable amount of energy is spent in thermal cooling or inefficient power supplies causing electrical losses), these issues are present regardless of queries being forwarded and do not affect the performance numbers comparisons.

The power P absorbed by a server is a function of its utilization level u [4], which depends on its *busy time* (time spent performing computations). Since B_j's servers are dedicated to process queries, in our scenario a server's busy time accounts for the amount of time needed to process all the queries assigned to the machine. If Q is the set of queries assigned to a server then the utilization level $u(Q)$ equals:

$$u(Q) = \frac{\sum_{q \in Q} s_q}{\Delta t} \ , \tag{6}$$

where s_q is the time required to solve a query q in Q, with $0 \leq s_q \leq \tau$.

Power consumption increases linearly with its utilization and it reaches its peak $\hat{P}$ when $u(Q) = 1$. If a server is idle, i.e. $u(Q) = 0$, it consumes only a small fraction of its peak power $\alpha\hat{P}$, with $0 < \alpha \leq 1/2$ [2]. Therefore, the power consumption of a server as a function of its utilization level $u(Q)$ can be written as:

$$P(Q) = \alpha\hat{P} + (1-\alpha)\hat{P}u(Q) \ . \tag{7}$$

At each time slot Δt we consider the energy E_j consumed by the backend B_j to be the sum of the electricity required by its m_j servers. Assuming B_j being composed by homogeneous machines, for all j we have:

$$E_j = \sum_{k=1}^{m_j} P(Q_{jk})\Delta t \ , \tag{8}$$

where Q_{jk} is the set of queries processed by the k-th server in B_j. By combining Equations 6 and 7, we have that:

$$\begin{aligned} E_j &= \alpha m_j \hat{P}\Delta t + (1-\alpha)\hat{P}\sum_{k=1}^{m_j} u(Q_{jk})\Delta t \\ &= \alpha m_j \hat{P}\Delta t + (1-\alpha)\hat{P}\sum_{k=1}^{m_j}\sum_{q \in Q_{jk}} s_q \ . \end{aligned} \tag{9}$$

We define Q_j as the full set of queries processed by the j-th backend's machines, i.e., $|Q_j| = y_j$. Finally, the average energy E_j consumed by backend B_j during a time slot Δt can be expressed as a function of the mean processing time μ as:

$$E_j = \alpha m_j \hat{P}\Delta t + (1-\alpha)\hat{P}\mu y_j = I_j + H_j y_j \tag{10}$$

The quantity I_j represents the energy consumed by B_j over Δt seconds when all its servers are idle. Instead, the quantity H_j represents the energy consumed to process queries.

Cost Model. We assume that every data center receives a limited amount of green energy per time slot, denoted as G_j. This green energy is available to the data center because it has its own green power plant, or because of special agreements between the search engine company and the energy provider. If G_j is not sufficient to satisfy the energy

consumption E_j the data center needs to buy additional, possibly brown, energy from the market, denoted as M_j.

The number of queries that, on average, can by processed by backend B_j by only exploiting green energy is:

$$g_j = \max\left(0, \frac{G_j - I_j}{H_j}\right) = \max\left(0, \frac{G_j - \alpha m_j \hat{P}\Delta t}{(1-\alpha)\hat{P}\mu}\right). \quad (11)$$

We consider that each data center D_j can consume energy available in two different prices per energy units. We denote as p_j^M the unitary cost of market energy at site D_j. This price varies throughout the day, typically being higher at daytime than during night [24]. Similarly, we define p_j^G to be the price of green energy at data center D_j. We assume the green energy to be more convenient than the market energy [13, 28]. Therefore, we would like to use only green energy to operate the backends and to process search requests, to reduce the search engine operational cost and carbon footprint. However, when G_j units of energy have been used in a given time period, we must resort to buy M_j units of market energy, to power the data center D_j for the remaining of the time slot. Finally, the operational costs C_j of a data center D_j can be expressed as:

$$C_j = \begin{cases} p_j^G E_j & \text{if } E_j \leq G_j \\ p_j^M E_j - (p_j^M - p_j^G)G_j & \text{otherwise} \end{cases} \quad (12)$$

Our goal is to minimize the overall operative cost in running a geographically distributed search engine, that is:

$$\min \sum_{j=1}^{N} C_j \,. \quad (13)$$

4. THE MIN COST FLOW QUERY FORWARDING ALGORITHM

In this section we propose and discuss a new query forwarding algorithm, called Min Cost Flow (MCF). This algorithm is executed periodically at the frontends of the data centers to decide if an incoming query must be processed locally or forwarded to another data center's backend in order to minimize the overall operational cost of a geographically distributed search engine. During every time interval Δt, we need to know how many queries each data center D_i must process locally or forward to data centers D_j, i.e., we need to compute values for the x_{ij} variables introduced in Section 3. In the following, we will show how these can be obtained by solving an instance of the Minimum Cost Flow Problem [6].

Algorithm 1 illustrates the MCF query forwarding algorithm. Firstly, each data center D_i must be able to locally compute the x_{ij}'s values for building a *forwarding table* X_i, such that $X_i[D_j] = x_{ij}$. This table is build every Δt seconds (lines 1–7). In order to generate this table, the algorithm needs an estimate of the query volumes a_i arriving to each data center in the next time slot Δt. The algorithm also requires per-site information related to its maximum sustainable query volume V_i, the available green energy G_i and the energy prices (p_i^M, p_i^G) at that time interval. We assume that data centers exchange messages every Δt seconds containing all those values [15].

At this point, whenever a query q arrives to the frontend F_i, the data center estimates its required processing time s_q (line 9). Next, F_i uses its forwarding table X_i to decide

```
1  every Δt seconds do
2  |  A = {a_1, ..., a_N}
3  |  V = {V_1, ..., V_N}
4  |  Γ = {G_1, ..., G_N}
5  |  Π = {p_1^M, ..., p_N^M, p_1^G, ..., p_N^G}
6  |  X_i = GenerateForwardingTable(A,V,Γ,Π)
7  end

8  forall the incoming queries q ∈ Q do
9  |  s_q = EstimateProcessingTime(q)
10 |  J = {D_j : X_i[D_j] > 0 ∧ s_q + RTT(i,j) ≤ τ}
11 |  if J is empty then
12 |  |  process q locally
13 |  else
14 |  |  D_j = select D_j ∈ J in a round robin fashion
15 |  |  X_i[D_j] ← X_i[D_j] − 1
16 |  |  if X_i[D_j] = 0 then
17 |  |  |  remove D_j from X_i
18 |  |  end
19 |  |  forward q to D_j
20 |  end
21 end
```

Algorithm 1: The MCF query forwarding algorithm.

to which sites it could forward a query to (line 10) and the frontend selects in a round-robin fashion a backend B_j from X_i, such that $X_i[D_j] > 0$ (line 14). This decision takes into account the expected round trip time from F_i to B_j. If the sum of the expected query processing time and the expected round trip time is smaller than the time budget τ, then q is forwarded to B_j (line 19). The value $X_i[D_j]$ is decreased by one and j is removed from X_i as soon as $X_i[D_j] = 0$ (lines 15–17). It is possible that no remote backend can process the query within the time budget; in such case the query is processed locally by default (lines 11–12).

Generating a forwarding table. The forwarding table X_i used in Algorithm 1 is obtained by solving an instance of the Minimum Cost Flow Problem (MCFP) [6] derived from the model of the distributed search engine discussed in Section 3.

The flow starts from some source nodes and every unit of flow needs to reach one or many sink nodes. In our case, queries represent the flow circulating in the network. Nodes are used to represent flow balance equations, i.e., the sum of incoming flow must be equal to the sum of outgoing flow. Every edge is labeled with three values representing the amount of flow passing through the link, its total capacity and cost per unit of flow.

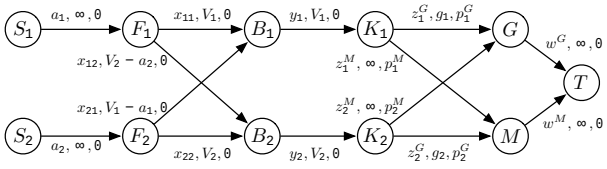


Figure 2: The MCFP instance to minimize the operational cost of a geographically distributed search engine.

Figure 2 illustrates how the operational cost minimization problem with two data centers can be represented as a MCFP instance. Source nodes are the query sources $S_1, \ldots, S_N$

described in Section 3. These nodes are connected to their correspondent frontends $F_1, \ldots, F_N$, using edges with infinite capacity and zero cost. Each source node S_i originates a_i units of flow (i.e., queries). It is not known, a priori, how many queries will arrive on frontend F_i during a particular time slot Δt. For this reason, a_i is estimated based on previous query arrivals (this process is detailed in Section 5).

Each frontend F_i is connected to every backend B_j. The arcs among frontends and backends have no cost. However they have different capacities. Edges (F_i, B_i) have capacity V_i, i.e., the maximum volume of queries per second sustainable by backend B_i (see Equation 4). Instead, edges (F_i, B_j) with $i \neq j$ have capacity $\lceil \max(0, V_j - a_j)/(N-1) \rceil$, as per Equation 5. The flows on the (F_i, B_i) edges represent the values for the x_{ij} variables defined in our model.

An artificial node K_j is introduced for every backend B_j, connected through an edge with zero cost and capacity V_j. These nodes enforce the constraint $y_j \leq V_j$ for every backend B_j (see Equation 4). The nodes K_j are connected to two more nodes, namely *Green* (G) and *Market* (M). The first node models the query processing by using green energy, while the second node by using market energy. Each K_j node is connected to the *Market* node through edges with cost p_j^M and infinite capacity. Moreover, each K_j node is connected to the *Green* nodes by edges with cost p_i^G but limited capacity g_i, i.e., the number of queries that B_i can process by using just green energy (see Equation. 11). The flows z_j^G and z_j^M are just an artifact included to satisfy the flow balance equations $y_j = z_j^G + z_j^M$.

Finally, the *G* and *M* nodes are connected to the sink node T. The edges ending in T have zero cost and infinite capacity and the flows w_G and w_M are an artificial mean to guarantee that the flow entering in T is equal to the flow leaving the sources S_i, i.e., $\sum_{i=1}^{N} a_i = w_G + w_M$.

Note that every edge but (C_i, G) and (C_i, M) has zero cost. Therefore, to minimize the flow cost indirectly minimizes the cost function defined by Equation 13.

Instances of the MCFP can be solved in polynomial time by using linear programming tools. More importantly, while MCF runs at every query arrival, the GENERATEFORWARDINGTABLE function needs to be executed less frequently (every Δt seconds).

Estimating the query processing time. There are several approaches to estimate the processing time of a query *before* executing it in the search engine backend. Documents relevant to a query are retrieved from the inverted index, by exhaustively traversing the posting lists relative to the query terms. In this case, processing time relates to the posting list lengths of the query terms [20]. Dynamic pruning techniques, such as MaxScore [30], can be deployed to speed up query processing. These techniques skip over portions of the posting lists by avoiding the evaluation of non relevant documents. When dynamic pruning is applied, queries with the same number of terms and similar posting list lengths can have widely different processing times and query efficiency predictors can be used [19].

Algorithm 1 is agnostic to the particular ESTIMATEPROCESSINGTIME() implementation. In the experimental section, we assume that this function uses an oracle which is employed in both the baseline and new algorithm.

5. EXPERIMENTAL SETUP

In this section, we determine empirically the potential of the proposed model and algorithm for reducing the operational cost of a distributed search engine without negatively impacting on its efficiency. In particular, we address three research questions, as follows:

1. What is the impact of the proposed approach and the baselines on the quality of service of the search engine?
2. Does our approach diminish the data centers' carbon footprint?
3. Do our MCFP-based model and the query forwarding MCF algorithm achieve energy cost savings comparable with reasonable baselines?

To answer questions 1, we measure the number of approximated and failed queries produced by the search engine. To answer question 2, instead, we need to evaluate the goodness of system in exploiting green energy. In fact, when the search engine misses the opportunity to consume green energy, it turns to the energy market – buying possibly brown energy and increasing its carbon footprint. For this reason we measure the system *green energy e ciency*, defined as the ratio between the amount of used green energy and the amount of green energy globally available to the search engine. Finally, to answer research question 3, during our simulations we measure the overall electricity expenditure of the search engine.

In the remainder of this section we define the experimental setup to address our research questions covering the baselines, the data centers, the data, the energy prices and the workload estimates.

Baselines. We will compare these results against two baselines. The first baseline, called NoForwarding, represents a standard multi-center Web search engine. It does not perform any query forwarding: the queries received by a frontend are processed in the local backend.

The second baseline, PROB, forwards the queries following the approach in [15]. The technique works as follows. At each query arrival, the data center D_i estimates the workloads (i.e., incoming queries) of each data center. Then, D_i looks at which data center is underloaded, i.e., which site has the opportunity to process forwarded queries. If these other data centers have a lower energy price, D_i simulates to redistribute its own workload towards these sites. However, D_i conservatively assumes that other data centers will try to do the same. Therefore, D_i equally divides such forwarding opportunity with its "competitors". The remaining queries are simulated to be processed locally. At the end of the simulation, the data center D_i has computed how many queries x_{ij} it would forward to data center D_j. A probability distribution is generated accordingly to these values and the incoming query is forwarded to a data center following such distribution.

Search data centers. Our experiments simulate a real distributed Web search engine with six data centers: DC-A, DC-B, DC-C, DC-D, DC-E and DC-F. We assume that these data centers are located in the capital cities of six different countries. In order to not disclose sensitive information, the countries are not revealed. We approximate network latencies between frontends and backends by considering the

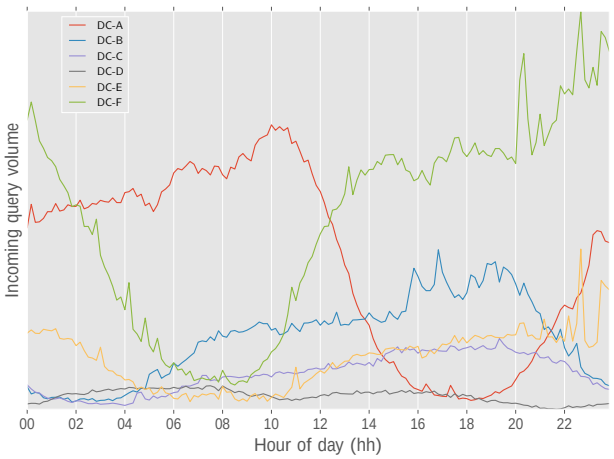


Figure 3: Query workload for six different Yahoo frontends during 24-hours.

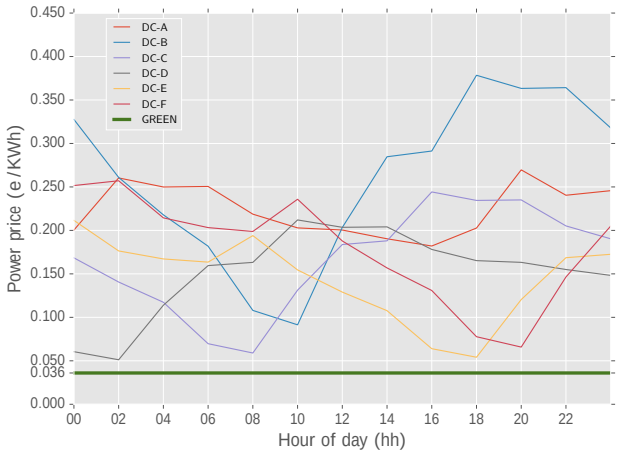


Figure 4: Market electricity prices and green energy price.

speed of light on copper wire (200,000 km/s) and the bird-fly distances between the relative cities [8, 15].

We assume that the data centers' backends contain identical servers. In this work, we experiment with two different kinds of server, i.e., we variate the α parameter defined in Section 3. The first server type represents normal servers which consume half of their peak power when idle (i.e., $\alpha = 0.5$). The other type represents next-generation servers, which are more energy-proportional, with $\alpha = 0.25$ [2].

The number of machines in a backend is determined by taking the reciprocal of the backend capacity in Equation 3. The maximum sustainable query volume, V_i, is set to be the 99.95th-percentile of the observed query loads, taken every second over the logs.

Data. We use real-world queries and arrival times sampled – in the order of tens of millions of points – from six different frontends of the Yahoo Web search engine, spanning one week. The first day of the log (see Figure 3) is used to tune our approach and the baselines. The remaining days are used in the simulation, i.e., queries are sent to the simulated data centers following the same load and arrival times reported in the log.

The processing times, on the other hand, are randomly sampled from an empirical distribution. This distribution is obtained by processing one million of unique queries using Terrier [23]. The platform is hosted on a dedicated Ubuntu 14.04 server, equipped with 32GB RAM and an Intel i7-4770K processor. Queries are taken from the MSN 2006 log, while documents relevant to such queries are retrieved from the TREC ClueWeb09 collection (cat. B). The corpus is indexed removing stopwords and stemming terms. The Elias-Fano schema is used for compression [32], and the resulting inverted index is kept in main memory. At retrieval time, dynamic pruning is applied by using the MaxScore algorithm [30]. The experimental mean processing time is 100 ms and its 99th-percentile is 589 ms. During our simulations, the query time budget τ is fixed to the reasonable value of 500 ms. This threshold has been chosen to avoid a drop ratio greater than 0.5%. We note that this value can be re-scaled if we consider shards with different sizes, for instance, when each machine index stores a lower amount of Web pages.

Electricity prices. We also need to set the price of green and market energy available in each data center, along with the amount of green energy available at each location.

Market electricity price varies by country and by hour of the day. To simulate this behavior, we generate a market electricity cost configuration in each data center. This configuration follows the price fluctuations observed over one week on the United Kingdom day-ahead market.[4] Prices are modified to reflect the cost of energy in the countries where the data centers are placed.[5] Also, prices are shifted accordingly to the timezone of the corresponding data center country. One day of the resulting electricity costs is shown in Figure 4. All times are normalized to the Greenwich Mean Time, i.e., GMT+0.

We assume that data centers have their own green power plants. Considering construction and maintenance costs, these produce energy for 0.036 e/KWh, which is cheaper than market energy [28]. As stated in Section 3, we assume that each data center D_i receives a fixed amount G_i of green energy at every time slot Δt. This quantity covers a fraction of the maximum amount of energy that D_i can possibly consume in Δt seconds. The maximum energy consumption of a data center is given by the product of the number m_i of its servers times their peak power consumption $\hat{P}$ times the time slot length Δt. Therefore, at each time slot, a datacenter D_i receives $G_i = \gamma m_i \hat{P} \Delta t$ KWh of green energy. We denote by γ the *green energy availability ratio* and range this value from 0 to 1 in the experiments. Green energy availability can fluctuate over the day. For instance, solar and wind energy production is susceptible to weather conditions. While our experiments do not consider this aspect, MCF can deal with variable green energy availability.

Workload estimate. Both PROB and MCF algorithms need to accurately estimate the workloads of data centers $a_1, \ldots, a_n$ in every time slot, to decide where to forward queries. Fol-

[4] `http://www.nordpoolspot.com` accessed 01-06-2015
[5] `https://en.wikipedia.org/wiki/Electricity_pricing`

Table 1: Time windows size (in seconds) for estimating the incoming query workloads of different data centers.

	$\alpha = 0.25$		$\alpha = 0.5$	
γ	PROB	MCF	PROB	MCF
0.0	200	40	200	40
0.1	200	40	190	40
0.2	200	40	200	40
0.3	200	40	200	40
0.4	200	40	190	40
0.5	200	1	210	40
0.6	200	1	190	40
0.7	190	10	200	1
0.8	200	1	200	10
0.9	200	1	200	1
1.0	200	10	200	10

lowing [15] we assume that data centers exchange messages about their current workload every second (i.e., $\Delta t = 1s$). This also corresponds to the execution frequency of the GenerateForwardingTable function (see Algorithm 1). In the succeeding time slot, each data center estimates the workloads of remote sites by using their recent history. Conservatively, a future workload is approximated to the maximum query volume observed over a time window for that remote data center [15]. In practice this implies that we need to select a time window size such that the workload estimates are as accurate as possible. As a working example consider a data center DC-A that is highly loaded. If the window size is too small and the query load briefly drops, we incur in the risk of forwarding too many queries to DC-A erroneously assuming that it is now lightly loaded. This situation will result in an high number of failed queries. On the other hand, if the time window is too large and DC-A is actually becoming lightly loaded, we will mistakenly produce low forwarding rates assuming that DC-A is still highly loaded. This implies that the algorithm will effectively miss the opportunity to save costs. Therefore, we use the first day of query log to determine a reasonable time window size. We set this size to the minimum possible value such that the failed query rate remains below the 0.5% threshold, as in [15]. The ideal time windows are reported in Table 1 and consistently used for the experiments. Note that MCF needs smaller time windows than PROB to estimate the incoming workloads.

6. RESULTS

In this section, we report on a comprehensive evaluation of MCF and PROB.The reported results are presented as percentage improvements over the NoForwarding baseline. We compare consistently the NoForwarding, PROB and MCF algorithms with varying configurations, namely the idle server power consumption fraction α and the data center green energy availability γ. We assume $\alpha = 0.5$ for current servers and $\alpha = 0.25$ for next-generation servers, while we vary γ from 0 (data centers completely powered by market energy) to 1 (data centers completely powered by green energy).

We aim to determine if our proposed MCF algorithm allows to markedly reduce market power consumption and operational costs with latency comparable to that achieved by the baselines. In particular, we firstly discuss the impact of the MCF approach on the quality of service of the system, measured in terms of the amount of approximated and failed queries. Then we evaluate the eco-friendliness of our algorithm with respect to the baselines, measured in terms of their green energy efficiency. Finally, we address our key research question concerning the cost savings MCF can obtain. The outcome of the experiments is summarized in Table 2.

Quality of service. The two first main columns of Table 2 report the values concerning the degradation in effectiveness of the different approaches, when queries are being processed in a distributed fashion across different data centers.

Results state that our proposed approach does not negatively impact the overall service quality of the multi-site Web search engine. Across all the tested green availability ratios γ, the MCF algorithm reduces the number of approximated queries. Moreover, thanks to query forwarding, approximated queries can be reduced from 1% to 11% percent with respect to NoForwarding, when currently existing servers (i.e., $\alpha = 0.5$) are utilized. Similar values are observable when the data center is equipped with more energy-proportional servers ($\alpha = 0.25$).

Table 2 also shows that MCF achieves the best absolute result in reducing the approximated queries with respect to the PROB baseline when a large fraction of the data centers is powered by green energy. However, while the PROB baseline maintains an almost constant decrease of $\sim$11% of approximated queries across all green energy availability configurations, MCF does not forward many queries for high green energy availability values, since a large quantity of cheap green energy is available locally at each data center. Consequently, data centers may incur in local overload situations that force them to early terminate some queries.

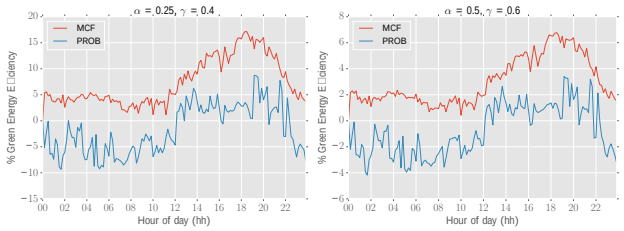
Note that we do not report the number of failed queries since in all configurations this value is always below the 0.5% threshold imposed in Section 5. Similarly, we do not report here the query response times, as the query time budget τ is fixed to 500 ms.

Green energy efficiency. As seen in Section 5, green energy efficiency gives us a measure of the search engine eco-friendliness. Table 2 shows that query forwarding plays a limited role in improving the green energy efficiency of a multi-site search engine built of current technology servers when the amount of green energy is limited ($\gamma \leq 0.5$). This effect happens because half of the energy consumed by the data centers is used just to keep the servers operative and idle, without processing any query. Therefore, there is no opportunity for processing requests using green energy when $\gamma \leq 0.5$. Similarly, MCF and PROB behave like NoForwarding also when green energy is abundant, i.e. $\gamma \geq 0.8$; in this case, in fact, a query is processed using green energy whether it is forwarded or not. With current servers, MCF shows eco-friendliness when $0.5 < \gamma < 0.8$, up to 1.66% when $\gamma = 0.6$. On the other hand, PROB proves to be less green energy efficient than NoForwarding. In fact, PROB is unaware of green energy availability and forwards queries where market energy is less expensive. While doing this, PROB misses the opportunity to exploit locally available green energy. Furthermore, it will not forward queries to data centers where market energy is costly but green energy is available.

Green energy efficiency improves when employing more energy-proportional servers. If $\alpha = 0.25$, MCF becomes

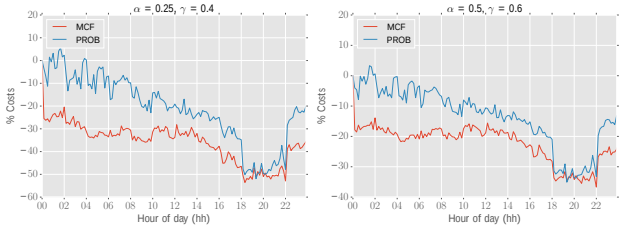
Table 2: Percentage of (a) Approximated queries, (b) green energy efficiency improvements, and (c) cost savings with respect to NoForwarding. The best results are reported in **bold**.

	(a) Approximated queries				(b) Green energy efficiency				(c) Cost			
	$\alpha = 0.25$		$\alpha = 0.5$		$\alpha = 0.25$		$\alpha = 0.5$		$\alpha = 0.25$		$\alpha = 0.5$	
γ	PROB	MCF	PROB	MCF	PROB	MCF	PROB	MCF	PROB	MCF	PROB	MCF
0.0	-11.15	-10.70	-11.16	-10.84	0.00	0.00	0.00	0.00	-9.86	-9.45	-4.71	-4.50
0.1	-10.95	-10.77	-11.08	-10.85	0.00	0.00	0.00	0.00	-11.96	-11.55	-5.39	-5.17
0.2	-11.20	-10.89	-11.03	-10.80	0.00	0.00	0.00	0.00	-15.44	-14.82	-6.30	-6.07
0.3	-11.03	-10.85	-10.82	-10.60	-2.58	0.26	0.00	0.00	-16.84	-19.52	-7.64	-7.34
0.4	-11.23	-11.46	-11.35	-10.95	-4.07	3.93	0.00	0.00	-9.02	**-25.51**	-9.75	-9.30
0.5	-11.16	-10.08	-11.11	-10.84	-2.19	**5.37**	0.00	0.00	-1.35	-21.73	-13.21	-12.67
0.6	-10.98	-7.77	-11.03	-11.21	0.02	1.14	-1.76	**1.66**	-1.23	-5.75	-5.52	**-15.79**
0.7	-11.19	**-11.73**	-10.98	-8.48	0.06	0.10	-0.35	1.22	-0.49	-0.66	-0.37	-6.16
0.8	-11.07	-3.21	-10.88	**-11.37**	0.00	0.00	0.03	0.04	-0.04	-0.17	-0.21	-0.28
0.9	-10.99	0.85	-11.16	-1.25	0.00	0.00	0.00	0.00	-0.04	-0.20	-0.01	-0.08
1.0	-11.07	-6.83	-11.20	-7.01	0.00	0.00	0.00	0.00	-0.03	-0.09	-0.00	-0.04



(a) (b)

Figure 5: Percentage of green energy efficiency improvements w.r.t. NoForwarding on a daily scale, over the first day of the test query log, for (a) $\alpha = 0.25, \gamma = 0.4$ and (b) $\alpha = 0.5, \gamma = 0.6$.



(a) (b)

Figure 6: Percentage of cost savings w.r.t. NoForwarding on a daily scale, over the first day of the test query log for (a) $\alpha = 0.25, \gamma = 0.4$ and (b) $\alpha = 0.5, \gamma = 0.6$.

more effective in exploiting green energy, improving over a larger range of green energy availability $0.3 < \gamma < 0.8$, up to 5.37% when $\gamma = 0.5$. Larger effects can be noticed on a smaller timescale, as highlighted in Figure 5. Instead, the PROB baseline proves to be inadequate in reducing the carbon footprint of a search engine, even when energy-proportional servers are employed.

Cost savings. Both MCF and PROB algorithms can help reducing the energy operational cost of a multi-site search engine, as shown in Table 2). When current servers are used, MCF and PROB similarly behave until the green energy availability $\gamma \leq 0.5$. Indeed, half of the energy available has to be consumed just to keep them operative. However, in order to be able to process queries, data centers need to buy additional energy from the energy market leaving no opportunity for MCF to exploit green energy for query processing. In any case, even when $\gamma \leq 0.5$, both PROB and MCF forward queries to sites with cheaper market energy, successfully reducing the operational costs of the whole search engine. As highlighted in Figure 7, when green energy availability is $0.5 < \gamma < 0.8$, MCF reduces the energy operational cost up to almost 16% w.r.t NoForwarding. Such savings are even higher if we look at daily variations, as shown in Figure 6.

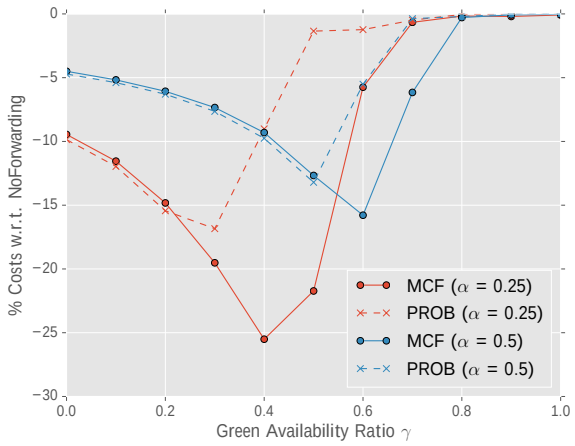


Figure 7: Percentage of cost savings w.r.t. NoForwarding, for different values of α and γ.

Conversely, PROB reduces the operational cost by only less than 6% when $0.5 < \gamma < 0.8$. This effect happens again because PROB is not able to use the information about the green energy availability, while MCF is able to exploit this knowledge. When $\gamma \geq 0.8$, little differences can be see with NoForwarding as green energy becomes highly available at every sites, and query forwarding does not make any difference. If we use more energy-proportional servers, larger benefits are observable over a wider range of green energy

availability. MCF reduces the energy expenditure of NoForwarding by more than 25% when just 40% of green energy is available. Again, these benefits are even larger when considering smaller time scales, as per Figure 6. Under the same configuration, PROB can only save less than 10%.

Finally, it is important to highlight that MCF obtains the best results when $0.3 < \gamma < 0.8$. This reinforces the importance of our results, as data centers would probably work with limited amounts of green energy due to its susceptibility to external variables such as the weather conditions.

7. CONCLUSIONS

In this paper we propose a novel model of a multi-center Web search engine that characterizes the frontend and backend components of the system, together with their processing capacity, to represent the energy operational cost of the search engine in presence of green energy resources. Using this model, we propose MCF, a novel query forwarding algorithm based on a Minimum Cost Flow Problem formulation. MCF is able to leverage the different pricing and availability of brown and green energy to reduce the energy expenditure of search systems. At the same time, MCF effectively takes into account the various backend processing capacities to maintain an acceptable service quality of the system.

We simulate the proposed algorithm using real query workloads obtained from six frontends of a live search engine and realistic electric price data. We compared our results with two baselines: NoForwarding, which represents a standard multi-site search engine; and the state-of-the-art PROB algorithm, which greedily forwards queries to the cheapest site following a estimated probabilistic distribution. We showed that with current technology servers, query forwarding plays a limited role in reducing the carbon footprint of search engines, although when a data center is equipped with next generation hardware query forwarding is a promising and effective technique to exploit green energy usage. This further reinforces the need for energy-proportional hardware in Web search engine data centers. Finally, we illustrated how MCF performs better than PROB in achieving cost savings, obtaining energy expenditure reductions that range from ∼15% to ∼25% with respect to the NoForwarding configuration. These savings are possible when limited quantity of green energy are available at the different remote sites, stating the importance of the MCF algorithm since green energy is typically attainable only in limited quantities due to external variables like weather conditions among others. For future work, we will consider more complex scenarios where different green power sources like solar and wind are available, and they vary dynamically during a day at the different sites as well as the development of more complex query workload forecast predictors.

8. ACKNOWLEDGEMENTS

We would like to thank Ana Freire for her useful comments and help in the early stages of this paper.

9. REFERENCES

[1] R. Baeza-Yates, A. Gionis, F. Junqueira, V. Plachouras, and L. Telloli. On the feasibility of multi-site web search engines. In *Proc. of CIKM*, 2009.

[2] L. A. Barroso, J. Clidaras, and U. Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan & Claypool Publishers, 2nd edition, 2013.

[3] L. A. Barroso, J. Dean, and U. Hölzle. Web search for a planet: The google cluster architecture. *IEEE Micro*, 23(2):22–28, 2003.

[4] L. A. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 40(12):33–37, 2007.

[5] R. Blanco, B. B. Cambazoglu, F. P. Junqueira, I. Kelly, and V. Leroy. Assigning documents to master sites in distributed search. In *Proc. of CIKM*, 2011.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, USA, 2004.

[7] B. B. Cambazoglu, V. Plachouras, and R. Baeza-Yates. Quantifying performance and quality gains in distributed web search engines. In *Proc. of SIGIR*, 2009.

[8] B. B. Cambazoglu, E. Varol, E. Kayaaslan, C. Aykanat, and R. Baeza-Yates. Query forwarding in geographically distributed search engines. In *Proc. of SIGIR*, 2010.

[9] G. Chowdhury. An agenda for green information retrieval research. *Inf. Process. Manage.*, 48(6):1067–1077, 2012.

[10] European Commission - Joint Research Centre. The European Code of Conduct for Energy Efficiency in Data Centre.

[11] The Climate Group for the Global e-Sustainability Initiative. Smart 2020: Enabling the low carbon economy in the information age, 2008.

[12] A. Freire, C. Macdonald, N. Tonellotto, I. Ounis, and F. Cacheda. A self-adapting latency/power tradeoff model for replicated search engines. In *Proc. of WSDM*, 2014.

[13] I. Goiri, K. Le, M. Haque, R. Beauchea, T. Nguyen, J. Guitart, J. Torres, and R. Bianchini. GreenSlot: Scheduling Energy Consumption in Green Datacenters. In *Proc. of SC*, 2011.

[14] F. P. Junqueira, V. Leroy, and M. Morel. Reactive Index Replication for Distributed Search Engines. In *Proc. of SIGIR*, 2012.

[15] E. Kayaaslan, B. B. Cambazoglu, R. Blanco, F. P. Junqueira, and C. Aykanat. Energy-price-driven Query Processing in Multi-center Web Search Engines. In *Proc. of SIGIR*, 2011.

[16] J. G. Koomey, C. Belady, M. Patterson, and A. Santos. Assessing trends over time in performance, costs, and energy use for servers, Intel, 2009.

[17] K. Le, R. Bianchini, T. D. Nguyen, O. Bilgir, and M. Martonosi. Capping the brown energy consumption of internet services at low cost. In *Proc. of GREENCOMP*, 2010.

[18] D. Lo, L. Cheng, R. Govindaraju, L. Barroso, and C. Kozyrakis. Towards Energy Proportionality For Large-Scale Latency-Critical Workloads. In *Proc. of ISCA*, 2014.

[19] C. Macdonald, N. Tonellotto, and I. Ounis. Learning to Predict Response Times for Online Query Scheduling. In *Proc. of SIGIR*, 2012.

[20] A. Moffat, W. Webber, J. Zobel, and R. Baeza-Yates. A Pipelined Architecture for Distributed Text Query Evaluation. *Inf. Retr.*, 10(3):205–231, 2007.

[21] U.S. Department of Energy. Quick start guide to increase data center energy efficiency, 2009.

[22] A. C. Orgerie, M. D. d. Assuncao, and L. Lefevre. A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Comput. Surv.*, 46(4):47:1–47:31, 2014.

[23] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma. Terrier: A High Performance and Scalable Information Retrieval Platform. In *Proc. of OSIR Workshop*, 2006.

[24] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs. Cutting the Electric Bill for Internet-scale Systems. In *Proc. of SIGCOMM*, 2009.

[25] F. B. Sazoglu, B. B. Cambazoglu, R. Ozcan, I. S. Altingovde, and O. Ulusoy. A financial cost metric for result caching. In *Proc. of SIGIR*, 2013.

[26] E. Schurman and J. Brutlag. Performance related changes and their user impact. In *Proc. of Velocity*, 2009.

[27] F. Silvestri. Mining query logs: Turning search usage data into knowledge. *Found. Trends Inf. Retr.*, 4(1–2):1–174, Jan. 2010.

[28] C. Stewart and K. Shen. Some Joules are More Precious Than Others: Managing Renewable Energy in the Datacenter. In *Proc. of HotPower Workshop*, 2009.

[29] A. Teymorian, O. Frieder, and M. A. Maloof. Rank-energy selective query forwarding for distributed search systems. In *Proc. of CIKM*, 2013.

[30] H. Turtle and J. Flood. Query Evaluation: Strategies and Optimizations. *Information Processing & Management*, 31(6):831–850, 1995.

[31] U.S. Department of Energy. Best Practices Guide for Energy-Efficient Data Center Design.

[32] S. Vigna. Quasi-succinct Indices. In *Proc. of WSDM*, 2013.