Heterogeneity-aware Workload Placement and Migration in Distributed Sustainable Datacenters

Dazhao Cheng*, Changjun Jiang† and Xiaobo Zhou*

*Department of Computer Science, University of Colorado, Colorado Springs, USA
†Department of Computer Science & Technology, Tongji University, Shanghai, China
Email addresses: dcheng@uccs.edu, cjjiang@tongji.edu.cn, xzhou@uccs.edu

***Abstract*—While major cloud service operators have taken various initiatives to operate their sustainable datacenters with green energy, it is challenging to effectively utilize the green energy since its generation depends on dynamic natural conditions. Fortunately, the geographical distribution of datacenters provides an opportunity for optimizing the system performance by distributing cloud workloads. In this paper, we propose a holistic heterogeneity-aware cloud workload placement and migration approach, sCloud, that aims to maximize the system goodput in distributed self-sustainable datacenters. sCloud adaptively places the transactional workload to distributed datacenters, allocates the available resource to heterogeneous workloads in each datacenter, and migrates batch jobs across datacenters, while taking into account the green power availability and QoS requirements. We formulate the transactional workload placement as a constrained optimization problem that can be solved by nonlinear programming. Then, we propose a batch job migration algorithm to further improve the system goodput when the green power supply varies widely at different locations. We have implemented sCloud in a university cloud testbed with real-world weather conditions and workload traces. Experimental results demonstrate sCloud can achieve near-to-optimal system performance while being resilient to dynamic power availability. It outperforms a heterogeneity-oblivious approach by 26% in improving system goodput and 29% in reducing QoS violations.**

I. Introduction

Today, major cloud service operators have taken various initiatives to operate their datacenters with renewable energy partially or completely [1]. Google, Facebook, and Apple have started to build their own green power plants to support the operation of their datacenters. Researchers envision that in the near future datacenters, at least micro-clouds, can be completely powered by renewable energy and be self-sustainable [17], [24]. Most green power plants use wind turbines and/or solar panels for power generation. Unlike traditional energy, the availability of green energy varies widely during the times of a day, seasons of the year, and geographical locations of the power plants. Such intermittency makes it very hard for sustainable datacenters to effectively use green energy.

At the same time, many large organizations operate multiple datacenters in a wide geographical region for several reasons, e.g., disaster tolerance and uniform access time for widely distributed clients. The power demands of these datacenters are highly dependent on the resource requirement of various hosted workloads. But the green power generations for distributed sustainable datacenters significantly vary over time and location. It makes the task of workload placement challenging, especially when the workload is heterogeneous. For example, most banks have to process the transactional applications for trading stocks and the batch workloads for analyzing the model of stock performance at the same time. Fortunately, the geographical distribution of the datacenters not only poses challenges but also opens up opportunities in the cloud workload management.

First, it is difficult to control the system power consumption under a dynamic power supply rather than under a static power budget. Most previous studies assume a static power budget as the constraint for workload management [8], [13], [14]. However, the green power supply in a self-sustainable datacenter highly depends on the natural weather conditions and is often time-varying. It is challenging to effectively match the power supply and demand in a self-sustainable datacenter.

Second, it is challenging to determine the locations where cloud workloads should be placed while their performance requirements are ensured. Datacenters are commonly shared by many users for quite different uses. An important trend is to co-locate heterogeneous workloads, i.e., transactional workloads and batch jobs, on the shared server infrastructure in datacenters for resource utilization efficiency [2], [4].

Transactional requests are relatively flexible to be dispatched to datacenters at different locations for workload management. However, a batch job's processing mostly relies on its data file which is typically not replicated across multiple datacenters. For example, a log analytic job is highly related to its local data and hard to be processed in other datacenters. There is an apparent difference between transactional requests and batch jobs in their data dependence. Most previous studies on distributed datacenter management [8], [18], [32] focus on the transactional workload and pay little attention to batch jobs. They may significantly affect the system performance and lead to serious QoS violations when workloads are heterogeneous.

Finally, the dynamic green power supply and geographical distribution of the datacenters also create opportunities for joint performance-power optimizations in distributed self-sustainable datacenters: (1) Although transactional workloads are flexible to be dispatched, their traffic intensities vary widely over time [7], [12]. Matching the computational loads

to the actual power supplies of different datacenters could achieve significant power savings. (2) Batch jobs can be migrated from one datacenter to another datacenter where the current power supply is relatively sufficient. (3) Given the dynamic power supply, a careful cloud workload placement and migration planning can maximize the overall system performance.

In this paper, we propose and develop a heterogeneity-aware cloud workload placement and migration approach, sCloud, that aims to maximize the system goodput in distributed self-sustainable datacenters. The key insight of sCloud is that the cloud workload, i.e., transactional requests and batch jobs, can be distributed to different locations based on their time-varying green power availabilities. Firstly, we model the intermittent generation of green energy to predict its production, with respect to the varying weather conditions at the geographical location of each datacenter. Then, we model the performance of transactional workload and batch workload with various resource allocations for each datacenter. Furthermore, we formulate the core objective of sCloud as a constrained optimization problem, in which the constraints capture the QoS requirements, the time-varying workloads and the intermittent availability of green power. Finally, We take advantage of batch job migration to reshape the power demand at different locations when the power supplies of geographically distributed datacenters vary widely. Specifically, our contributions are as follows:

- We propose a heterogeneity-aware cloud workload placement and migration approach in distributed self-sustainable datacenters. It maximizes the system performance in terms of system goodput, based on the time-varying green power supply, heterogeneous workload characteristics and QoS requirements.
- We design an optimization-based algorithm to dynamically place transactional requests to distributed datacenters, with respect to their green power supplies. In order to further improve the system goodput, we integrate another online algorithm to dynamically migrate batch jobs across distributed datacenters when the green power supplies vary widely at different locations.
- We have implemented sCloud in our university cloud testbed and performed extensive evaluations with real-world weather conditions and workload traces. Experimental results demonstrate sCloud achieves near-to-optimal system performance while being resilient to dynamic power availability. It outperforms a heterogeneity-oblivious approach GLB [18] by 26% in system goodput improvement and 29% in reducing QoS violations.

The rest of this paper is organized as follows. Section II reviews related work. Section III describes the design of sCloud. Section IV presents the optimization-based workload and resource management. Section V gives the testbed implementation. Section VI presents the experimental results and analysis. Section VII concludes the paper.

II. Related Work

As the environmental concerns and the energy consumption of datacenters continuously grow, developing sustainable datacenters is becoming an increasingly important mission for major Internet service operators [2], [8], [18], [22], [32].

The vast majority of the previous studies on the sustainable energy management has focused on the single datacenters. These studies aim to achieve sustainable operation driven by green energy supply partially or completely from following aspects: (1) Studies [2], [10], [16] focus on the energy demand side of a datacenter. (2) Studies [9], [15], [30] focus on matching energy supply of a datacenter server cluster with its energy demand. (3) Studies [26], [29] focus on different energy storage approaches in the sustainable datacenters to improve their green energy usage efficiency.

Aksanli *et al.* [2] designed an adaptive job scheduler to increase the green energy usage in a sustainable datacenter, which utilizes short term prediction of solar and wind energy production. This scheduling method may violate the QoS requirement due to unnecessarily delaying batch jobs. Goiri *et al.* proposed GreenHadoop [10], a MapReduce framework for a datacenter powered by solar energy and using electrical grid as a backup. It aims to maximize the green energy consumption by job scheduling. It does not consider the potential opportunities to improve the green energy usage by workload distribution across distributed datacenters.

A few recent studies start to utilize green energy in distributed datacenters. Deng *et al.* [8] proposed an adaptive request routing approach to meet the operational cost, QoS, and carbon footprint goals. Zhang *et al.* [32] proposed GreenWare, a middleware system that dynamically dispatches transactional requests to distributed datacenters to maximize the use of green energy within the allowed operation budget. However, these studies only consider transactional requests that are of low cost for routing and do not consider another important category of cloud workload, i.e., batch jobs.

Recently, Liu *et al.* proposed GLB [18], a geographical load balancing approach that can significantly reduce the required capacity of green energy by using the energy more efficiently with request dispatching. GLB provides a representative workload dispatching and capacity provisioning method to minimize the system energy cost and the request delay cost. Our work differs from this effort in that we consider the workload heterogeneity and batch job migration across distributed datacenters.

III. sCloud Design

sCloud is a holistic workload and resource manager that maximizes the system goodput in the presence of dynamic green power supply. The key insights are transactional workload can be dynamically dispatched to distributed datacenters in order to reshape the power demand of each datacenter, and batch jobs, e.g., MapReduce, can be migrated across datacenters to further improve system performance and green energy usage. sCloud considers to meet the QoS requirements

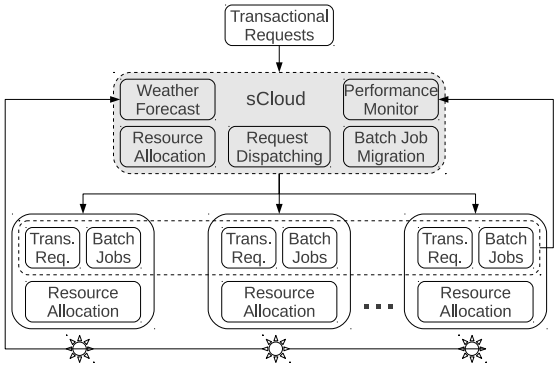


Fig. 1. The architecture of sCloud.

of heterogeneous workloads while making dispatching and migration decisions.

Figure 1 shows the architecture of sCloud. In each control interval, sCloud performs following three tasks:

- Collects the weather forecast information at each geographical location and predicts its available green power.
- Monitors the performance of applications hosted in each datacenter and the intensity of heterogeneous workloads.
- Decides how to place the transactional requests to geographically distributed datacenters, allocate the available resource to heterogeneous workloads in each datacenter, and migrate batch jobs across datacenters.

We first propose system goodput, a unified metric to quantify the system performance with heterogeneous workloads. We then formulate the workload management and resource provisioning as a constrained optimization problem.

A. Quantifying System Performance

Although heterogeneous workloads have individual measures of client-perceived performance, such as request response time and job completion time, a unified metric is needed for cloud providers to quantify the benefit of resource allocation. We define system goodput as the total useful work delivered to users in a certain period of time. Specifically, it is the amount of effective data throughput completed by interactive requests or batch jobs that meet their corresponding service level objectives (SLOs). Similar metrics have been used to quantify system performance for interactive requests [11] and batch jobs [6], respectively.

Formally, we define system goodput $G_i(k)$ at the i_{th} datacenter in time interval k as the sum of two different workloads:

$$G_i(k) = G_i^t(k) + G_i^b(k), \tag{1}$$

where $G_i^t(k)$ and $G_i^b(k)$ are the goodput of transactional requests and batch jobs, respectively. For heterogeneous workloads, $G_i(k)$ is uniformly defined as:

$$G_i(k) = \frac{(\sum_{l=1}^{m} d_l) * \delta_i(k)}{\Delta t(k)}, \tag{2}$$

where d_l is the data size of task l in a group of m tasks that finish during time period k. $\Delta t(k)$ is the length of interval k. To count only useful work, decay function $\delta_i(k)$ discounts the data throughput from tasks with violated SLOs:

$$\delta_i(k) = \begin{cases} 1 & \text{if } t_i(k) < t_{soft}. \\ 1 - \frac{t_i(k) - t_{soft}}{t_{soft}} & \text{if } t_{soft} \le t_i(k) \le t_{hard}. \\ 0 & \text{if } t_i(k) > t_{hard}. \end{cases} \tag{3}$$

We use a SLO with two time bounds, t_{hard} and t_{soft}, in the decay function. While t_{hard} sets a hard deadline for task completion, beyond which no revenue is generated, t_{soft} is a soft deadline whose violation will incur reduction in revenue. We ensure that hard deadlines to be no longer than 2 times of the soft deadlines so that the decay function never becomes negative. Accordingly, $\delta_i(k)$ considers the data delivered by m tasks as useful work if the observed average finish time $t_i(k)$ meets t_{soft}. Violations of t_{soft} and t_{hard} result in a linear decay in the counted throughput and zero work, respectively.

For the transactional workload, we treat requests as individual tasks and estimate the transaction data size $d^t(l)$ as the size of the response sent back to the clients. For batch jobs, we use the input data size to approximate the task size $d^b(l)$. Typically, these jobs are split and processed by many sub tasks distributed across a number of computing slots in a cluster. If there are more tasks than the available slots, the tasks will be processed by multiple waves [28]. Then the SLO requirement is divided into the sub-task level as following:

$$n_{wave} = \lceil \frac{n_{task}}{n_{slot}} \rceil, \tag{4}$$

$$t_{task}^{SLO} = \frac{t_{job}^{SLO}}{n_{wave}}, \tag{5}$$

where n_{task}, n_{slot} and n_{wave} are the number of tasks, slots and waves for the batch job, respectively. The SLO requirement at the job level t_{job}^{SLO} is divided into task level t_{task}^{SLO}. Many batch jobs provide the interface to query such execution information, such as the *JobTracker* in a Hadoop environment.

B. Optimization Problem Formulation

The optimization problem of sCloud is formulated as,

$$\max \sum_{k=1}^{K} \sum_{i=1}^{N} [G_i^t(k) + G_i^b(k)], \tag{6}$$

$$\text{s.t. } P_i(k) = P_i^{solar}(k) + P_i^{wind}(k), \tag{7}$$

$$\lambda^t(k) = \sum_{i=1}^{N} \lambda_i^t(k), \forall i \in N, k \in K, \tag{8}$$

$$\lambda_i^t(k) \ge 0, \lambda_i^b(k) \ge 0, \forall i \in N, k \in K. \tag{9}$$

Objective Eq. (6) aims to maximize the system goodput. Constraint Eq. (7) represents that the power supply of the i_{th} datacenter is determined by the local green power generation amount. Eq. (8) represents that the overall transactional workload arrival rate is the accumulated value of that at each datacenter. This is a complex optimization problem with both

TABLE I
NOTATIONS.

Symbol	Meaning
N	Total number of datacenters
k	Control interval
P_i	The total power supply at the i_{th} datacenter
P_i^{solar}	The solar power supply at the i_{th} datacenter
P_i^{wind}	The wind power supply at the i_{th} datacenter
$\lambda^t(k)$	The system transactional workload arrival rate
$\lambda_i^t(k)$	Transactional workload arrival rate at the i_{th} datacenter
$\lambda_i^b(k)$	Batch workload arrival rate at the i_{th} datacenter

nonlinear objective function and constraints. To solve the problem, we first build the green energy model for power supply prediction. We then model the performance of heterogeneous workloads in each datacenter. Finally the optimization problem is transformed to a nonlinear programming problem. The notations used for problem formulation are listed in Table I.

IV. OPTIMIZATION-BASED MANAGEMENT

A. Green Power Prediction

In contrast to brown energy, solar and wind energy generation may not provide a reliable, consistent source of energy due to the time-varying weather conditions. We use prediction methods to estimate the amount of green energy in a given interval and utilize that data to make decisions of workload placement and resource provisioning in distributed datacenters.

We use the model introduced by the work in [23] to predict solar and wind power. The solar model is based on the simple premise that energy generation is inversely related to the amount of cloud coverage. It is expressed as: $P^{solar}(k) = B^{solar}(k)(1-CloudCover)$, where $P^{solar}(k)$ is the amount of solar power predicted for interval k, $B^{solar}(k)$ is the amount of solar power expected under ideal sunny conditions, and $CloudCover$ is the forecasted percentage cloud cover (given as a fraction between 0 and 1). We use historical data from NREL [19] to instantiate B^{solar}.

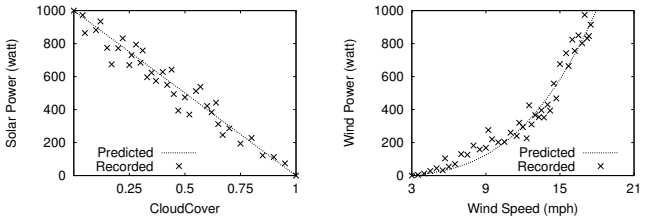
The wind power model is based on the cubic wind power production function [23]. That is, $P^{wind}(k) = a \times (v^w(k))^3 + b$, where v^w is the windspeed, a and b are parameters depending on the locations. We fit this power curve with the observed data from NREL [19] using the least-squares method to generate the wind power model. The parameters a and b for three locations used in our implementation are obtained as shown in Table II.

Given the data from NREL [19], we implement solar and wind energy prediction models at the granularity of 10 minutes. We assume that each sustainable datacenter has 7 solar panels and a micro-turbine with capability of producing $1KW$ respectively. Figure 3 demonstrates that the average difference between each observed and predicted value is small.

In this work, we focus on the CPU resource allocation since CPU is the major power contributor in datacenters [2], [10], [17]. There is a linear relationship between the available CPU resource and the power consumption [2], [8]. Thus, the power constraint in the optimization problem is transformed to the resource availability constraint.

TABLE II
WIND POWER MODEL PARAMETERS.

Locations	California	Hong Kong	Dublin
a	0.1732	0.1697	0.1783
b	3.1207	3.0641	2.9372



(a) Solar power model prediction. (b) Wind power model prediction.

Fig. 3. The accuracy of green power prediction.

B. Transactional Workload Placement

As in related studies [25], [32], we use the classic queueing theory to model the performance of heterogeneous workloads in each datacenter. sCloud includes transactional and batch workloads in the model to obtain a more realistic system view in the cloud environment.

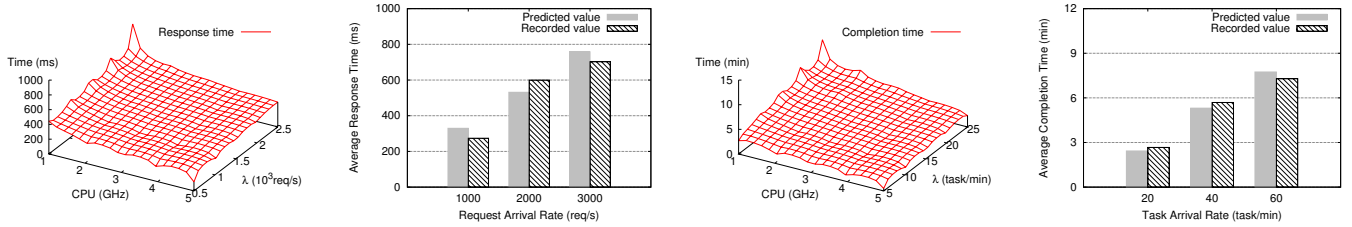
1) Transactional workload model: For the transactional workload, performance goals are typically defined in terms of the average response time [20] or throughput [11]. As in previous study [32], we use a multi-server queueing model M/M/c to model the performance of transactional workload in a datacenter. The average response time of the requests at the i_{th} datacenter is represented as

$$t_i^t = \frac{P_Q}{c\mu_i^t - \lambda_i^t} + \frac{1}{\mu_i^t}. \tag{10}$$

Here, t_i^t includes the waiting time $\frac{P_Q}{c\mu_i^t - \lambda_i^t}$ and the service time $\frac{1}{\mu_i^t}$. c is the number of servers and μ_i^t is the average service rate of a single server in the i_{th} datacenter. λ_i^t is the request arrival rate in the i_{th} datacenter. P_Q is the probability of requests waiting in the queue. Similar to others [21], [32], we assume that all servers will keep busy. Hence, we have P_Q equal 1.

2) Batch workload model: The performance of batch workload concerns the completion time of individual jobs. For the batch workload, schedulers, such as Fair Scheduler and Capacity Scheduler in Hadoop can share the resource among multiple jobs. We consider there are multiple jobs hosted in the cluster. As demonstrated in [25], [31], job arrivals can be modeled by a Poisson process, where the interarrival times are typically exponentially distributed. Thus, we model the batch workload at the task level by a M/GI/1/PS queueing system.

In a Hadoop cluster, each job is divided to multiple tasks for processing. Let t_i^b denote the average completion time of



(a) Average response time. (b) RUBiS Prediction error. (c) Average completion time. (d) Hadoop Prediction error.

Fig. 2. Performance model accuracy evaluation with different amount of resources and workloads.

Algorithm 1 The Algorithm of Placement.

Initial state $v(k)$, $k \leftarrow 1$;
repeat
  At the beginning of control interval k;
  Inputs:
  / Weather forecast: $CloudCover_i(k)$, $v_i^w(k)$;/
  / Workloads: $\lambda_i^b(k)$, $\lambda^t(k)$;/
  / Service rate: $\mu_i^t(k-1)$, $\mu_i^b(k-1)$;/
  Update the performance model parameters;
  Predict the available green power $P_i(k)$;
  Solve the problem by $fmincon$ function in MATLAB;
  Output: The optimal point $V^*(k) = \{\lambda_i^t(k), r_i^t(k), r_i^b(k)\}$;
until k = K

the batch tasks. Based on the queueing foundation, we have

$$t_i^b = \frac{1}{\mu_i^b - \lambda_i^b}. \tag{11}$$

Here λ_i^b is the workload arrival rate and μ_i^b is the average service rate that is determined by its resource allocation.

In order to evaluate the accuracy of the models, we implement RUBiS as the transactional workload and Gridmix2 of Hadoop as the batch workload. As shown in Figures 2(a) and 2(c), the average response time and the completion time vary when given different amount of resource and workload. The accuracy is measured by the normalized root mean square error (NRMSE), a standard metric for deviation. Figures 2(b) and 2(d) show that the real measured data and predicted data are very close, with the NRMSE 10.4% and 6.9% for RUBiS and Hadoop workloads, respectively.

3) Placement algorithm: Given the models above, the optimization problem described in Section III is transformed to a nonlinear programming problem. Let $V(k) = \{\lambda_i^t(k), r_i^t(k), r_i^b(k)\}$ denote the optimization variable. As shown in Algorithm 1, at the beginning of each control interval, sCloud takes current weather forecast, workload arrival rate and the service rate as inputs. It generates the optimal transactional workload placement solution and resource allocations at each datacenter in the current interval to maximize the system goodput.

We use the $Karush\text{-}Kuhn\text{-}Tucker$ conditions [3] to determine the optimal point $V^*(k)$ for the placement solution $\lambda_i^t(k)$ and resource allocations $r_i^t(k), r_i^b(k)$. We implement the proposed algorithm based on a standard nonlinear programming solver, $fmincon$, which is provided in the optimization toolbox of MATLAB.

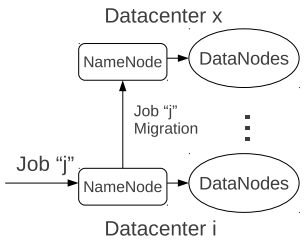


Fig. 4. Batch job migration.

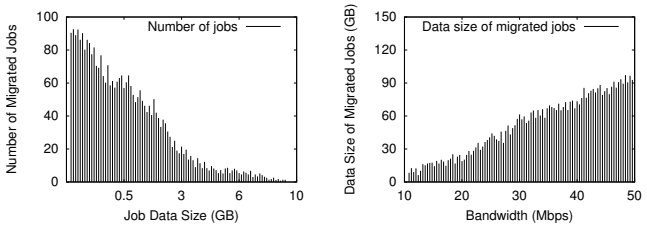
C. Batch Job Migration

1) Why migrate: Although the transactional workload placement can effectively distribute the workload to geographically distributed datacenters, job migration provides another opportunity to further reallocate the workload according to the power availability at individual datacenters. When the power supply is too low for one datacenter, as Algorithm 1 shows, sCloud does not dispatch any transactional requests to the datacenter. Moreover, sCloud may further migrate jobs from this datacenter to other datacenters to meet the QoS requirement and to maximize the system goodput.

2) Where to migrate: Algorithm 2 determines which job should be migrated and where to migrate. If no new transactional request is placed to a datacenter in the k_{th} interval, the algorithm is initialized when a new batch job is submitted to the datacenter i. If job migration is needed, the batch job is migrated to another datacenter before it starts running at the source datacenter. Figure 4 illustrates a scenario where a batch job j is submitted to the datacenter but there is no sufficient power at the datacenter to run the job in addition to the currently running jobs and meet their QoS requirements. sCloud performs two steps to decide where to migrate job j.

Firstly, sCloud selects a destination candidate x from the potential destination datacenters. It is done by comparing the completion time of the job j among the possible migration solutions if j would be migrated from the current datacenter to others. Note that batch jobs typically rely on their data files and the job migration incurs additional cost for data transfer.

Specifically, the estimated completion time of the migrated



(a) Data size impact. (b) Bandwidth impact.

Fig. 5. Data size and bandwidth impact on job migration.

job j includes the data transfer delay and the job process time at the x_{th} datacenter. The delay caused by data transfer between the source datacenter i and the destination datacenter x is represented as

$$t^d_{i,x}(j) = \frac{DataSize_j}{Bandwidth_{i,x}}, (i, x \in \{1, \cdots, N\}). \quad (12)$$

Here, delay $t^d_{i,x}$ is determined by the bandwidth between two datacenters and the amount of data that needs to be transferred. The amount of data is determined by specific job input and output data size. We focus on the batch jobs with large scale input data. Figure 5 depicts the statistical results of batch job migration in our experiment using FaceD workload [10], [31]. Figure 5(a) shows that the migration of the small jobs is more frequent than that of the large jobs. Figure 5(b) demonstrates that the data size of migrated jobs is proportionally increased when the bandwidth increases.

The job process time at the x_{th} datacenter is estimated as

$$t^{process}_x(j) = t^b_x \times n_{wave}, \quad (13)$$

where t^b_x is the average task completion time in the x_{th} datacenter based on the performance model described by Eq. 11. n_{wave} is the number of waves for job j, which can be obtained from the *JobTracker* in a Hadoop environment.

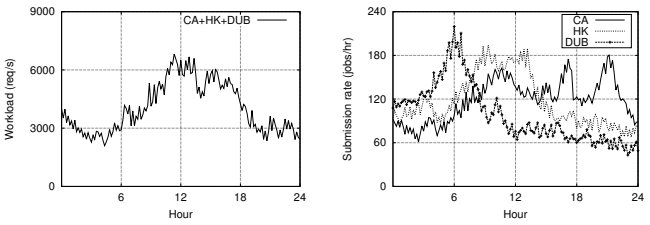
Secondly, the algorithm makes decision of job migration based on the performance comparison between the destination candidate x and the current datacenter i. If the completion time at the current datacenter is longer than that at the destination candidate, job j is migrated to the destination candidate. Otherwise, job j is still hosted in the current datacenter.

Algorithm 2 is implemented as a daemon program at each datacenter, i.e., $NameNode$ of Hadoop cluster.

3) How to migrate: sCloud uses the *DistCp* in a Hadoop environment to transfer the job data from the source datacenter to the destination datacenter. *DistCp* is a tool used for large inter-cluster data copying. It is implemented as a MapReduce job where the work of copying is done by the maps that run in parallel across the cluster. It tries to give each map approximately the same amount of data, at least 256 MB. Note that the *DistCp* expects the data transfer between the Hadoop clusters to be of the same version.

Algorithm 2 The Algorithm of Job Migration.

1: Search from the first datacenter, $i \leftarrow 1$;
2: **repeat**
3: **if** $\lambda^t_i(k) = 0$, Job j submits **then**
4: /* Find the best destination candidate x*/
5: $x = \arg\min_x [t^d_{i,x}(j) + t^{process}_x(j)], x \in \{1, \cdots, N\}$
6: /* Make decision of job migration*/
7: **if** $t^{process}_i(j) > [t^d_{i,x}(j) + t^{process}_x(j)]$ **then**
8: Job j is migrated from i to x
9: **end if**
10: **end if**
11: **until** $i = N$



(a) Transactional workload. (b) Batch job submission rate.

Fig. 6. The transactional and batch workloads traces at CA, HK and DUB.

V. System Implementation

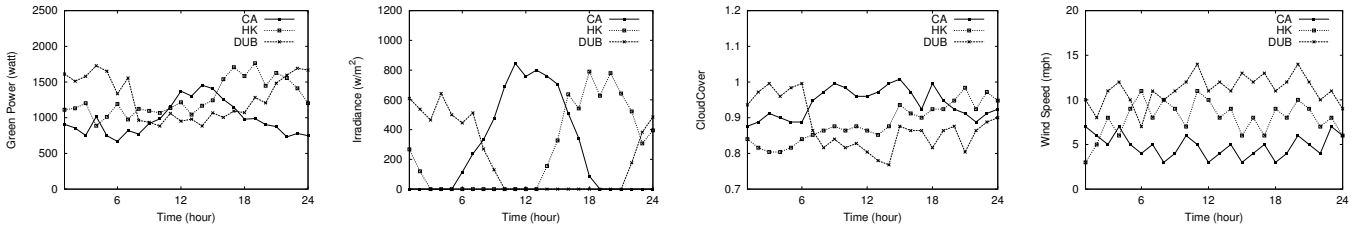
A. Testbed

We built a testbed in a university prototype datacenter, which consists of 108-core CPUs and 704 GB memory. VMware vSphere 5.0 is used for server virtualization. VMware vSphere module controls the CPU usage limits in MHz allocated to the virtual machines (VMs). It also provides an API to support the remote management of VMs.

Large corporations built their distributed datacenters at multiple regions worldwide, e.g., North America, Asia and Europe. We built three clusters based on the testbed to mimic three self-sustainable datacenters located at California (CA), Hong Kong (HK) and Dublin (DUB), respectively. The average bandwidth between the clusters is 27Mbps by measurement. Each cluster includes a number of transactional Web servers and a Hadoop cluster. The Hadoop used in the experiment is 1.0.3 version and configured with 11 VMs, i.e., 1 $NameNode$ and 10 $DataNode$. Fair Scheduler is used in the experiment. As in the work [28], each DataNode is configured with a single map and reduce slot. Each VM is allocated 1 VCPU and 1 GB memory. All VMs use Ubuntu Server 10.04 with Linux 2.6.32.

B. Real-world Workloads

For the transactional workload, we use open-source RUBiS as the benchmark and a real Internet trace from Wikipedia.org [27] to mimic the daily dynamics of workload volume. The trace represents the users' behavior in visiting the Wikipedia website. Figure 6(a) shows the transactional workload used in the experiments. The number of concurrent users dynamically changes from 2700 to 6300 in 24 hours. Our experiments set the soft response time bound to be 1000 ms and the hard response time bound to be 1500 ms [20].



(a) Green power supply. (b) Sun irradiance. (c) Could cover. (d) Wind speed.

Fig. 7. The local green power supply, sun irradiance, cloud cover and wind speed at California (CA), Hong Kong (HK) and Dublin (DUB).

For the batch workload, we use a synthetic workload, "Facebook-Derived (FaceD)", which models Facebook's production workload [10], [31]. FaceD contains jobs with widely varying execution times and data set sizes, representing a scenario where the cluster is used to run many different types of batch applications. By default Hadoop configuration, we submit all jobs with normal priority in the experiments.

Table III shows the FaceD workload characteristics and the SLO soft and hard completion time bounds. We do not run the Facebook code itself. Rather, we mimic the characteristics of the jobs using "loadgen". Loadgen is a configurable MapReduce job from the Gridmix2 benchmark in the Hadoop distribution. Figure 6(b) shows a real-world trace from Facebook [5] to mimic the submission rate of jobs.

C. Green Power Supply and Weather Conditions

We use the green power data from the National Renewable Energy Laboratory (NREL) database [19]. This database contains time series data in 10-minute intervals from more than 30,000 measurement points worldwide. In experiments we picked three measurement points at CA, HK and DUB to get real green power traces. Figure 7(a) shows the amount of green power supply varies over time during one day at CA, HK and DUB. The time used in the figures is Pacific Time.

To emulate the intermittent availability of green energy, we use meteorological data from the Measurement and Instrumentation Data Center of NREL [19]. As shown in Figure 7, a variety of meteorological data, including sun irradiance, cloud cover, and wind speed, is covered in those records from the NREL. Prior studies [2], [32] have shown that the data from the NREL is quite accurate in weather prediction.

TABLE III
FACED WORKLOAD CHARACTERISTICS.

% Jobs	# Maps	# Reds	Data (GB)	SLO (min) (soft-hard)
59	4	1	0.36	1-1.5
9.8	10	1-2	0.69	2-3
8.7	20	1-5	1.65	5-7.5
8.5	40	2-10	2.60	10-15
5.7	80	4-20	5.00	15-22.5
4.4	150	8-38	9.38	30-45
2.5	300	15-75	18.75	50-75
1.3	600	30-150	37.5	100-150

D. SCLOUD Components

1) sCloud Controller: We implement sCloud control algorithm on a separate VM, which issues commands to the virtualized server cluster using VMware vSphere API 5.0. The algorithm invokes a nonlinear programming solver $fmincon$ in the optimization toolbox of MATLAB.

2) Power Monitor: The real-time power consumption of the virtualized cluster is measured at the resource pool level. The power monitor gathers the measurement data through VMware ESX 5.0 Intelligent Power Management Interface sensors.

3) Performance Monitor: For the transactional workload, it uses a sensor program provided by RUBiS client for performance monitoring in terms of request response time and the data size of each request. For MapReduce batch jobs, it measures each task completion time and task size by $JobTracker$ on the $NameNode$ periodically.

4) Resource Allocator: It uses vSphere API to impose CPU usage limits on the VMs. The vSphere module provides an interface to execute a method $ReconfigVM$ to modify a VM's CPU usage limit, ranging from 0 to 2.8 GHz.

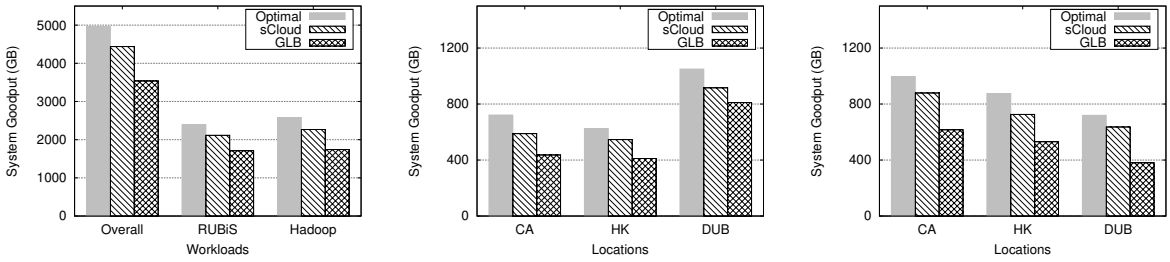
VI. PERFORMANCE EVALUATION

We first demonstrate the system goodput achieved by sCloud. We then illustrate the effectiveness of self-optimizing workload placement with dynamic green power supply. Furthermore, we show the self-adaptiveness of power consumption control and resource allocation by sCloud. Finally, we show the effectiveness of batch job migration control and the power prediction impact on sCloud.

For reference, we obtained the optimal system goodput based on an offline optimization process using the workload trace and power supply trace. Note this offline training based optimization is an ideal but not practical solution. For illustration, we implemented GLB [18], an online migration-oblivious geographical load balancing approach for cloud workload. For fairness, we tailored GLB so that its available resource for provisioning is also dynamically driven by the power supply.

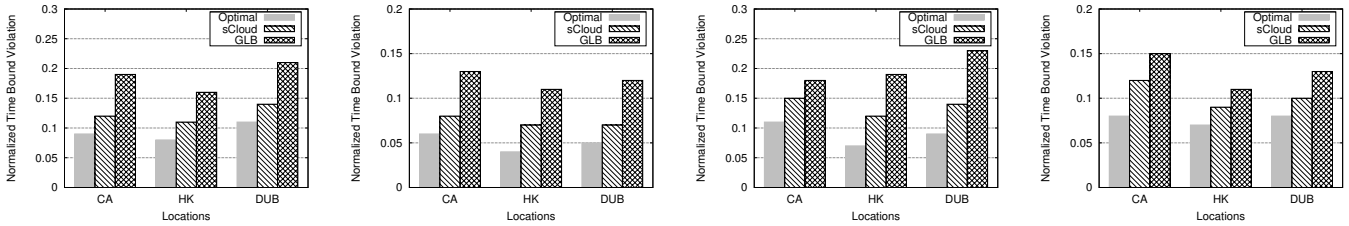
A. System Goodput Improvement

Figure 8 compares the system goodput achieved by Optimal, sCloud and GLB approaches in a one-day green power supply scenario. Figure 8(a) shows that the overall system goodput of sCloud is 26% more than that achieved by GLB. And sCloud achieves near-to-optimal performance, obtaining 91% system



(a) Overall system goodput. (b) RUBiS workload. (c) Hadoop workload.

Fig. 8. The system goodput improvements in a one-day scenario at California (CA), Hong Kong (HK) and Dublin (DUB).



(a) RUBiS soft violations. (b) RUBiS hard violations. (c) Hadoop soft violations. (d) Hadoop hard violations.

Fig. 9. Soft and hard QoS requirement violations of RUBiS and Hadoop workloads at California (CA), Hong Kong (HK) and Dublin (DUB).

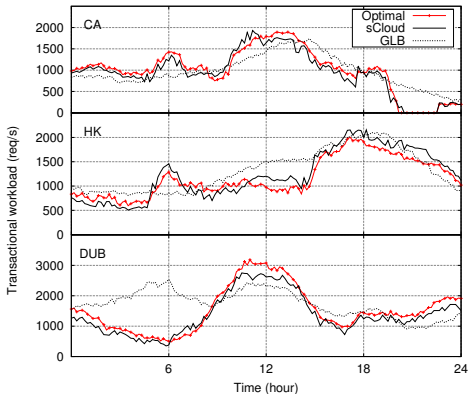


Fig. 10. RUBiS workload placement.

goodput of the optimal solution. For the goodput of RUBiS and Hadoop workloads, sCloud outperforms GLB by 23% and 30% while achieving 92% and 89% of Optimal, respectively.

Figures 8(b) and 8(c) show the performance comparison of three different approaches for RUBiS and Hadoop workloads at different locations. Figure 8(b) shows that the goodput of RUBiS workload by sCloud outperforms that by GLB 24%, 27% and 17% at CA, HK and DUB, respectively. And it achieves 92%, 94% and 90% of the optimal solution at CA, HK and DUB, respectively. Figure 8(c) shows that the goodput of Hadoop workload by sCloud outperforms that by GLB 24%, 27% and 17% at CA, HK and DUB, respectively. And it achieves 92%, 94% and 90% of the optimal solution at CA, HK and DUB, respectively.

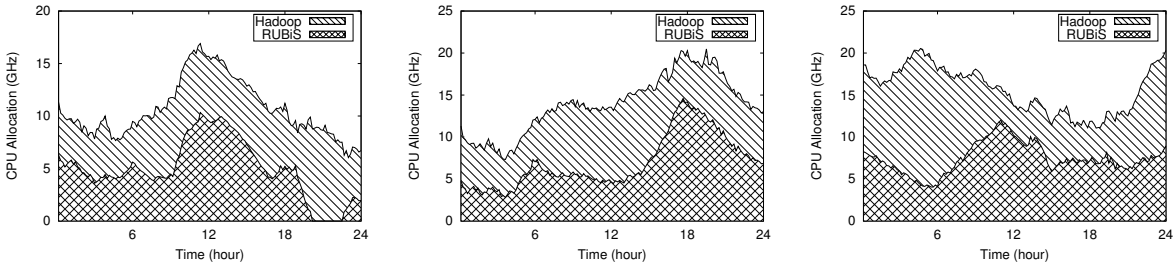
Figure 9 compares the QoS time bound violations of RUBiS workload and Hadoop workload by Optimal, sCloud and GLB approaches in the one-day scenario. Figures 9(a) and 9(b) show that sCloud significantly reduces the soft response time bound violation and hard response time bound violation of RUBiS workload by 39% and 43% respectively. Figures 9(c) and 9(d) show that sCloud also effectively reduces the soft completion time bound violation and hard completion time bound violation of Hadoop workload by 35% and 29% respectively.

B. Self-optimizing Placement

Figure 10 provides a microscopic view of the dynamic RUBiS workload placement by different approaches in the one-day scenario. It depicts how to place RUBiS requests to CA, HK and DUB by Optimal, sCloud and GLB. In the first eight-hour stage, GLB dispatches more requests to DUB due to more green power supply at DUB. But when the batch workload at DUB is relative high between the 5_{th} and 7_{th} hours, sCloud and Optimal allow more requests to be placed to CA and HK to improve the overall system goodput. This is due to the fact that sCloud and Optimal take both transactional and batch workloads into account.

In the second stage, the green power supplies of three locations are similar with each other. sCloud and Optimal place more RUBiS workload to DUB than to others since it has the smallest batch workload. Because the batch workload at HK is highest during this period, sCloud and Optimal place smaller RUBiS workload to HK than to GLB.

In the final stage, sCloud and Optimal place more RUBiS workload to HK and DUB while GLB places the requests to three datacenters in proportion to their power supplies. This is because the green power supplies at HK and DUB are more than that at CA. During the 20_{th} to 22_{th} hours, sCloud and Optimal place all transactional requests to HK and DUB to



(a) Resource allocation at CA. (b) Resource allocation at HK. (c) Resource allocation at DUB.

Fig. 11. Self-adaptiveness of resource allocation by sCloud at CA, HK and DUB.

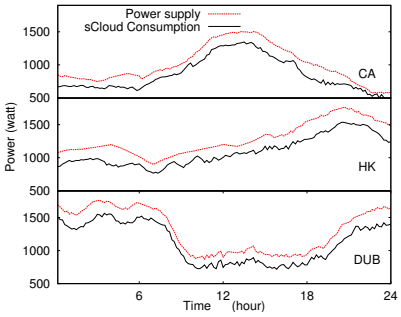


Fig. 12. Self-adaptiveness of power consumption control.

improve the system goodput. GLB still places RUBiS requests to three datacenters in proportion to their power supplies, losing the opportunity for the system goodput optimization.

C. Resource and Power Control

1) Power consumption control: Figure 12 shows sCloud power consumptions respect the dynamic power supplies at CA, HK and DUB, respectively. We can observe that sCloud is able to control the power consumption by adaptively increasing or reducing the available resource at each datacenter. This is due to the obtained model in Section IV for green power prediction. sCloud applies a threshold-based power capping technique [9] to make sure that the real power consumption is below the real power supply.

2) Resource allocation control: Figure 11 shows the dynamic CPU resource allocations between RUBiS workload and Hadoop workload at CA, HK and DUB, respectively. It shows that the overall available resources are allocated in proportion to their green power supplies. Figure 11(a) shows the resource allocated for RUBiS at CA increases between the 11_{th} and 17_{th} hours as RUBiS workload increases. sCloud allows all resource to be allocated to Hadoop workload between the 20_{th} and 22_{th} hours. This is because the Hadoop workload at CA is relative high and sCloud places all RUBiS workload to HK and DUB during this period.

As shown in Figure 11(b), sCloud allocates more resource to RUBiS workload while Hadoop workload is low during last eight hours. Figure 11(c) shows at DUB datacenter, more resource is allocated to Hadoop workload during the beginning stage and the end stage compared to the middle stage. At the beginning stage, it is to satisfy the increased Hadoop workload at DUB datacenter. At the end stage, it is to deal with the batch jobs migrated from CA datacenter.

D. Batch Job Migration

Figures 13(a) and 13(b) show the number of batch jobs and the amount of their data migrated from CA to HK and CA to DUB between the 20_{th} and 22_{th} hours. The comparison between Figure 13(a) and 13(b) illustrates more batch jobs are migrated from CA to DUB than CA to HK. This is due to the fact that DUB has more green power supply than HK has in this period.

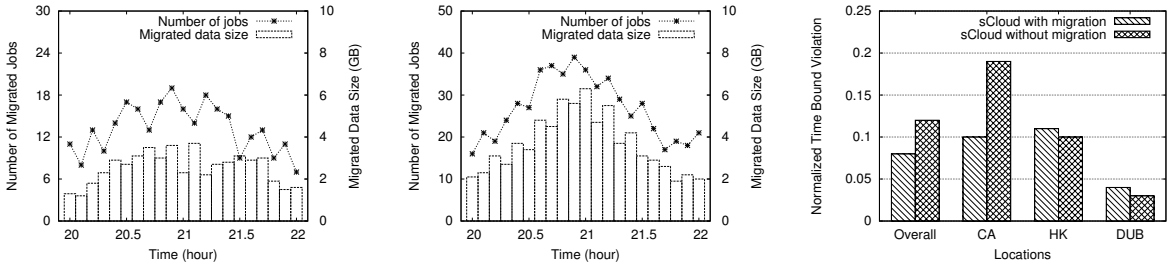
Figure 13(c) demonstrates that sCloud's job migration significantly reduces the job soft time bound violations at CA. Because additional batch jobs are migrated from CA to HK and DUB, the job completion time violations at HK and DUB are increased slightly. However, Figure 13(c) illustrates sCloud has improved the overall soft time bound violation of batch jobs between the 20_{th} and 22_{th} hours.

E. Power Prediction Impact

Figure 14 shows the impact of the green power prediction error on the system goodput and QoS violation improvements. We use the real power supply traces as the baseline to observe the effectiveness of sCloud under the various power prediction errors. Figure 14(a) shows that the system goodput improvement is reduced from 37% to -41% as the power prediction error increases from 0% to 35%. Figure 14(b) demonstrates both soft and hard time bound violations deteriorate significantly as the power prediction error increases.

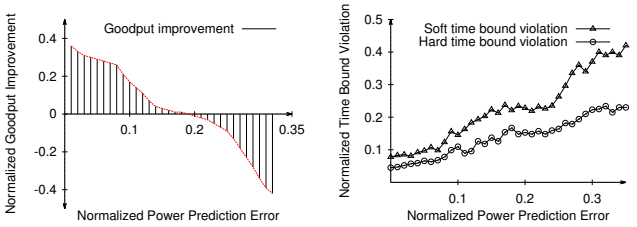
VII. Conclusion and Future Work

In this paper, we focus on the heterogeneous workload management in distributed self-sustainable datacenters. We have proposed and developed a self-optimizing cloud workload management approach, sCloud, that can improve the system goodput with respect to the dynamic green power supply. As demonstrated by the modeling, optimization and experimental results based on the testbed implementation, its main contributions are near-to-optimal performance, adaptive workload management and resilience to dynamic power availability. The main technical novelty of sCloud lies in the integration of request placement, dynamic resource allocation and batch job



(a) Job migration from CA to HK. (b) Job migration from CA to DUB. (c) Soft time bound violations.

Fig. 13. Batch job migration control and the job soft time bound violations improvement between the 20_{th} to 22_{th} hours.



(a) Goodput impact. (b) Violation impact.

Fig. 14. Power prediction error impact on performance improvement.

migration. sCloud can significantly improve the system performance and green energy usage by self-optimizing workload and resource management.

Our future work will integrate the energy storage technique with sCloud for sustainable computing in green datacenters.

ACKNOWLEDGEMENT

This research was supported in part by U.S. NSF CAREER award CNS-0844983, research grant CNS-1217979, and NSF of China research grant 61328203.

REFERENCES

[1] Various green datacenters. http://www.ecobusinesslinks.com/.
[2] B. Aksanli, J. Venkatesh, L. Zhang, and T. Rosing. Utilizing green energy prediction to schedule mixed batch and service jobs in data centers. In *Proc. USENIX HotPower*, 2011.
[3] S. Boyd and L. Vandenberghe. Convex optimization. 2004.
[4] D. Carrera, M. Steinder, I. Jordi Torres, and E. Ayguade. Autonomic placement of mixed batch and transactional workloads. *IEEE Trans. on Parallel and Distributed Systems*, 23(1), 2012.
[5] Y. Chen, S. Alspaugh, D. Borthakur, and R. Katz. Energy efficiency for large-scale mapreduce workloads with significant interactive analysis. In *Proc. ACM EuroSys*, 2012.
[6] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz. The case for evaluating mapreduce performance using workload suites. In *Proc. IEEE/ACM MASCOTS*, 2011.
[7] D. Cheng, Y. Guo, and X. Zhou. Self-tuning batching with dvfs for improving performance and energy efficiency in servers. In *Proc. IEEE/ACM MASCOTS*, 2013.
[8] N. Deng, C. Stewart, D. Gmach, M. Arlitt, and J. Kelley. Adaptive green hosting. In *Proc. ACM ICAC*, 2012.
[9] D. Gmach, J. Rolia, C. Bash, Y. Chen, T. Christian, A. Shah, R. Sharma, and Z. Wang. Capacity planning and power management to exploit sustainable energy. In *Proc. IEEE CNSM*, 2010.
[10] I. Goiri, K. Le, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini. Greenhadoop: Leveraging green energy in data-processing frameworks. In *Proc. ACM EuroSys*, 2012.
[11] Y. Guo, P. Lama, and X. Zhou. Automated and agile server parameter tuning with learning and control. In *Proc. IEEE IPDPS*, 2012.
[12] P. Lama, Y. Guo, and X. Zhou. Autonomic performance and power control for co-located web applications on virtualized servers. In *Proc. ACM/IEEE IWQoS*, 2013.
[13] P. Lama, Y. Li, A. Aji, P. Balaji, J. Dinan, S. Xiao, Y. Zhang, W. Feng, R. Thakur, and X. Zhou. Power-aware dynamic placement and migration in virtualized GPU environments. In *Proc. IEEE ICDCS*, 2013.
[14] P. Lama and X. Zhou. Ninepin: Non-invasive and energy efficient performance isolation in virtualized servers. In *Proc. IEEE/IFIP DSN*, 2012.
[15] C. Li, R. Zhou, and T. Li. Enabling distributed generation powered sustainable high-performance data center. In *Proc. IEEE HPCA*, 2013.
[16] S. Liu, S. Ren, Q. Gang, M. Zhao, and S. Ren. Profit aware load balancing for distributed cloud data centers. In *Proc. IEEE IPDPS*, 2013.
[17] Z. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, Z. Wang, M. Marwah, and C. Hyser. Renewable and cooling aware workload management for sustainable data centers. In *Proc. ACM SIGMETRICS*, 2012.
[18] Z. Liu, M. Lin, A. Wierman, S. Low, and L. Andrew. Geographical load balancing with renewables. In *ACM SIGMETRICS*, 2011.
[19] NREL. Measurement and instrumentation data center. http://www.nrel.gov/midc/.
[20] P. Padala, K.-Y. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant. Automated control of multiple virtualized resources. In *Proc. ACM EuroSys*, 2009.
[21] X. Rao, L.and Liu, L. Xie, and W. Liu. Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment. In *Proc. IEEE INFOCOM*, 2010.
[22] S. Ren and Y. He. Coca: Online distributed resource management for cost minimization and carbon neutrality in data centers. In *Proc. SC*, 2013.
[23] N. Sharma, J. Gummeson, D. Irwin, and P. Shenoy. Cloudy computing: Leveraging weather forecasts in energy harvesting sensor systems. In *Proc. IEEE CNSM*, 2010.
[24] R. Singh, D. Irwin, P. Shenoy, and K. Ramakrishnan. Yank: Enabling green data centers to pull the plug. In *Proc. USENIX NSDI*, 2013.
[25] J. Tan, X. Meng, and L. Zhang. Delay tails in mapreduce scheduling. In *Proc. ACM SIGMETRICS*, 2012.
[26] E. Thereska, A. Donnelly, and D. Narayanan. Sierra: Practical power-proportionality for data center storage. In *Proc. ACM EuroSys*, 2011.
[27] G. Urdaneta, G. Pierre, and M. van Steen. Wikipedia workload analysis for decentralized hosting. *Computer Networks*, 53(11), 2009.
[28] A. Verma, L. Cherkasova, and R. H. Campbell. Resource provisioning framework for mapreduce jobs with performance goals. In *Proc. ACM/IFIP/USENIX Middleware*, 2011.
[29] D. Wang, C. Ren, A. Sivasubramaniam, B. Urgaonkar, and H. K. Fathy. Energy storage in datacenters: What, where, and how much? In *Proc. ACM SIGMETRICS*, 2012.
[30] Y. Wang, R. Chen, Z. Shao, and T. Li. Solartune: Real-time scheduling with load tuning for solar energy powered multicore systems. In *Proc. IEEE RTCSA*, 2013.
[31] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica. Job scheduling for multi-user mapreduce clusters. In *TR UCB/EECS-2009-55, Berkeley*, 2009.
[32] Y. Zhang, Y. Wang, and X. Wang. Greenware: Greening cloud-scale data centers to maximize the use of renewable energy. In *Proc. ACM/IFIP/USENIX Middleware*, 2011.