# Providing Green SLAs in High Performance Computing Clouds

Md. E. Haque, Kien Le, Íñigo Goiri, Ricardo Bianchini, Thu D. Nguyen
{*mdhaque, lekien, goiri, ricardob, tdnguyen*}*@cs.rutgers.edu*
Department of Computer Science, Rutgers University

Technical Report DCS–TR–702, Dept. of Computer Science, Rutgers University, March 2013

***Abstract*—Demand for clean products and services is increasing as society is becoming increasingly aware of climate change. In response, many enterprises are setting explicit sustainability goals and implementing initiatives to reduce carbon emissions. Quantification and disclosure of such goals and initiatives have become important marketing tools. As enterprises and individuals shift their workloads to the cloud, this drive toward quantification and disclosure will lead to demand for *quantifiable green cloud services*. Thus, we argue that cloud providers should offer a new class of green service, in addition to existing (energy-source-oblivious) services. This new class would provide their clients with explicit service-level agreements (which we call Green SLAs) for the percentage of renewable energy used to run their workloads.**

**In this paper, we first propose an approach for High Performance Computing cloud providers to offer such a Green SLA service. Specifically, each client job specifies a Green SLA, which is the minimum percentage of green energy that must be used to run the job. The provider is penalized if it accepts the job but violates the Green SLA. We then propose (1) a power distribution and control infrastructure that uses a small amount of hardware to support Green SLAs, (2) an optimization-based framework for scheduling jobs and power sources that maximizes provider profits while respecting Green SLAs, and (3) two scheduling policies based on the framework. We evaluate our framework and policies extensively through simulations. Our main results show the tradeoffs between our policies, and their advantages over simpler greedy heuristics. We conclude that a Green SLA service that uses our policies would enable the provider to attract environmentally conscious clients, and especially those who require strict guarantees on their use of green energy.**

## I. Introduction

It is well known that datacenters consume an enormous amount of electricity. This consumption translates into high carbon emissions, since most of the electricity is produced using fossil fuels. A 2008 study estimated world-wide datacenters to emit 116 million metric tons of carbon, slightly more than the entire country of Nigeria [1]. With increasing societal awareness of these emissions and climate change, there is increasing demand for cleaner products and services.

In response, enterprises have started to set explicit sustainability goals and create initiatives to reduce their carbon emissions. Such efforts have become important marketing tools. For example, in 2012, the Global Reporting Initiative [2], a non-profit organization that seeks “to make sustainability reporting standard practice for all organizations,” registered 2,295 reports from companies such as AMD, Dell, and Microsoft. As enterprises and individuals shift their workloads to the cloud, this drive toward quantification and disclosure of the sustainability of business activities will lead to demand for *quantifiable green cloud services*.

A few small green cloud service providers, e.g., Green House Data [3], AISO [4], and GreenQloud [5], have already sprung up to meet this demand. However, it is difficult to quantify how green such providers are. For example, while two of Green House Data’s datacenters are 100% powered by renewable (wind) energy, a third (equipped with solar power) is not [6]. Furthermore, cloud clients may have different goals for their carbon emissions (e.g., 100% vs. 30% green energy). In fact, since not all clients would need a 100% green service, creating fully green systems unnecessarily increases costs.

Thus, instead of a one-size-fits-all approach, we argue that Infrastructure-as-a-Service (IaaS) cloud providers should offer a new class of service, with *explicit service-level agreements (SLAs) for the percentage of renewable energy used to run the clients’ workloads*. We refer to these renewable-energy SLAs as Green SLAs. Importantly, this class of service should not replace existing services (which are oblivious to energy sources). Rather, it would allow environmentally conscious clients to explicitly contract for use of green energy for their workloads. For example, a client interested in near-zero carbon emissions would contract for virtual machines (VMs) in its workload to run 100% on green energy. Others could contract for lower percentages of green energy. An enterprise could even contract for different SLAs over time, as their business goals and progress toward meeting those goals evolve.

Cloud providers can offer the above differentiated Green SLA service in different manners. For example, one approach is to account for the entire amount of green and brown (i.e., electrical-grid-sourced) energy consumed within an accounting period (e.g., a month) irrespective of which workload consumed how much of each type of energy. In this accounting approach, it suffices for the provider to show that enough green energy was used to meet all Green SLAs. Although simple, this approach would not suffice for some clients. Specifically, we expect that an increasing number of clients will require guarantees that their workloads are executed with specific fractions of green energy, especially if they are required to demonstrate so by legislation. For example, the UK government requires businesses consuming more than 6 GWh of brown energy per year to purchase carbon offsets from the market [7].

In this paper, we propose and evaluate a stricter paradigm for High Performance Computing (HPC) cloud providers, where a client’s Green SLA can only be satisfied by actually using green energy to run that client’s job (i.e., a collection of VMs). Specifically, we assume that each job submitted by a client specifies a desired Green SLA, in the form of a minimum percentage of green energy that must be used to run the job. The provider can accept or reject the job.[1] If it accepts a job, the provider earns a premium for the percentage of the client's job that must be run using green energy. However, the provider must pay a penalty if it violates the Green SLA. Meeting Green SLAs in the presence of intermittent sources of energy, such as solar and wind, is a challenging proposition.

As the cloud provider needs to differentiate green energy from brown energy to satisfy the Green SLAs, we assume that it operates a datacenter that either generates its own green energy (self-generation) or draws it directly from an existing nearby plant (co-location). In either scenario, the datacenter can also draw on brown energy as needed. (Importantly, note that we do *not* argue that self-generation or co-location will be the best approach for all sustainability-conscious datacenter operators. Rather, we argue that self-generation or co-location will be the approach of choice for many operators, as suggested by the many examples in [3]–[5], [8]–[10].)

However, having two distinct sources of energy (green and brown) is not enough to provide Green SLAs. The provider needs to bring the green energy all the way to the servers in the portion of the datacenter reserved to provide the Green SLA service. Thus, we propose a new power delivery infrastructure in which each rack reserved for the Green SLA service is dynamically switched between two energy sources, one entirely green, and one possibly a mixture of brown and green. The switching of the racks between the two energy sources is controlled by software running on a control module.

Next, we propose a software framework for optimization-based scheduling of client jobs and energy sources for the racks. We design two optimization-based scheduling policies using this framework. The policies seek to maximize the profit that the cloud provider can accrue by admitting and running clients' jobs. They predict the amount of green energy that will likely be produced in the future, and use these predictions together with jobs' execution information and Green SLAs to decide whether to admit jobs. They also generate schedules for executing the admitted jobs and for controlling the energy source of each rack. We also use two greedy heuristic scheduling policies as baselines for comparison.

Finally, we evaluate our proposed infrastructure and scheduling policies using simulation. Our main results demonstrate that HPC cloud providers can profitably offer a Green SLA service, where clients' Green SLAs can only be met by explicitly scheduling the clients' jobs to run on green energy. Further, our results show that the optimization-based policies can significantly outperform the greedy policies. Critically, policy parameters for the optimization-based policies can be adjusted to reflect different values placed on meeting Green SLAs. For example, the penalty can be set very high (even if the actual penalty the provider has to pay out to clients is lower) to force the policies to be conservative and avoid missing Green SLAs. Alternatively, the penalty can be set low to reflect a best effort environment, where violating green SLAs are not serious failures.

[1] The provider can also negotiate a Green SLA to reflect its commitments to other clients and the expected availability of green energy. We have not explored this scenario in the interest of simplicity.

In summary, this paper makes the following contributions: (1) we propose a new class of HPC cloud service that provides explicit SLAs for the use of green energy, (2) we propose a hardware infrastructure to support this new class of service; (3) we propose an optimization-based framework for admission control and scheduling to maximize profit while respecting green SLAs; (4) we use the framework to design two optimization-based policies; and (5) we evaluate our framework and policies extensively through simulation.

## II. Related Work

To our knowledge, this paper is the first to propose SLAs specifying the percentage of green energy to be used in the execution of datacenter workloads. Nevertheless, there have been related efforts in three topics, as we discuss next.

**Exploiting green energy in datacenters.** Many recent papers have focused on datacenters that exploit renewable energy [11]–[22]. This paper adds to this body of work. However, most of these works sought to maximize the use of green energy within performance bounds. In addition to this same goal, our work seeks to precisely assign the green energy to clients according to Green SLAs. We propose novel hardware and software to accomplish this assignment.

The most similar prior work is [12], which proposed to concentrate the green energy as much as possible on the servers used for the workloads of environmentally conscious clients. However, the authors did this concentration via strategically placed grid-tie devices, whereas we propose to have a separate power bus for green energy. Moreover, the authors did not consider workload migration or explicit Green SLAs.

**Managing workloads in green datacenters.** Researchers have proposed to schedule deferrable batch jobs to maximize the use of renewable energy [13], [14], [16]. In contrast, Krioukov *et al.* proposed to adjust the quality of the replies provided to users in non-deferrable interactive workloads [17]. For datacenters that run a mix of interactive and batch workloads, [11] and [20] proposed to adapt the amount of batch processing dynamically. Goiri *et al.* considered both deferrable and non-deferrable workloads [15]. Li *et al.* proposed two sets of servers to draw energy from two types of energy source (the electrical grid and a wind farm) [19]. Their work requires migrating workloads between the two sets to increase green energy usage. Our paper assumes non-deferrable batch workloads to mimic the increasingly common practice of renting VMs on demand to run HPC jobs. Moreover, our workload management for the first time involves carefully admitting and migrating VMs to meet Green SLAs.

**VM placement.** Researchers have addressed VM placement to achieve lower energy cost or fewer SLA violations [23]–[26]. These works either proposed migrating VMs across datacenters or carefully packing VMs within a datacenter to achieve lower energy consumption with bounded performance loss. None of these works considered green energy.

## III. Power Distribution Infrastructure

Figure 1 shows the power distribution and control infrastructure we propose for a module that would be used to support the Green SLA service. One or more of these modules can be

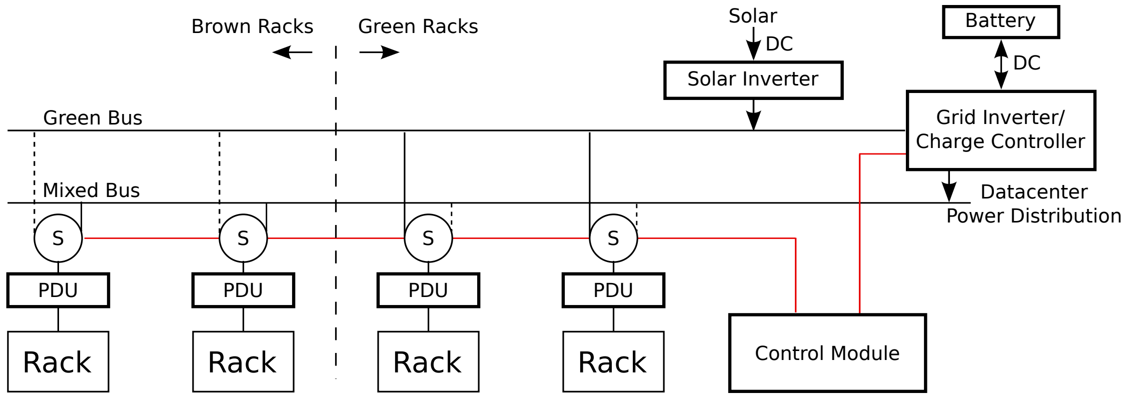



Fig. 1. Power distribution infrastructure. The vertical dash line shows an example partitioning of the racks into a brown part (where racks are switched to the Mixed Bus) and a green part (where racks are switched to the Green bus) by a scheduling policy. This scheduling is discussed in Section IV.

housed in a larger datacenter that also houses infrastructure for providing regular services (i.e., services to clients not interested in Green SLAs). Current IaaS providers already offer many classes of service, such as the Cluster Compute and Cluster GPU service classes of Amazon EC2.

The infrastructure contains two separate power buses, with the Green Bus carrying strictly green power and the Mixed Bus carrying a mix of brown and green power. Each rack is connected to a software-controlled transfer switch, which switches the power source for the rack between the Green and Mixed buses. The Grid Inverter/Charge Controller (GICC) is configured to charge the battery when there is excess green power, and discharge it when there is insufficient green power. Once the battery is full, excess green power is routed to the Mixed Bus. *The GICC is configured to never allow power flow from the Mixed bus to the Green bus or to the battery.*

The control module executes software that configures the GICC and the power switches. We detail this software in Section IV. The overall idea, however, is to use the solar power source, battery, and Green Bus to deliver *pure* green power to racks that can then be used to satisfy Green SLAs. This is why brown power is never allowed to enter the Green Bus and the battery. While excess green power may be routed to the Mixed Bus (so that it is not wasted), this green energy will not be counted toward any Green SLA.

Green energy is produced by a solar plant, either located on-site or at a nearby location. The solar source is supplemented by a *small* battery to smooth out any short-term variability in the solar power production (e.g., less energy is produced in a 15-minute interval than was predicted) as well as variability in power consumption. (Longer term matching of green energy and workload is done by the control software.)

In the proposed infrastructure, the solar inverter and GICC are both required for integrating a local green power source into the datacenter, independent of our proposal for a Green SLA service. The battery may or may not be needed, depending on whether the service provider wants to store excess green energy (vs. net-metering where available, or just waste excess green energy). Thus, the hardware cost for providing the Green SLA service is the extra power bus, the per-rack power switch, and possibly the small battery.

This design is based on our experience building an actual solar-powered micro-datacenter [15]; we have direct experience with all components, except for the software-controlled transfer switches. However, similar switches are available commercially and are commonly used in current datacenters for redundant power delivery [27], [28]. In addition, while this paper considers solar as the source of green energy, our framework can be easily applied to other sources of green energy, such as wind, as long as it is possible to predict the near-future production of the green energy.

## IV. Scheduling Framework

Given the above power distribution and control framework, our goal is to schedule client jobs, and brown and green energy consumption to maximize the cloud provider's profit while meeting the clients' Green SLAs. In this section, we first describe the proposed Green SLA service in more detail. We then overview the scheduler, formulate scheduling as an optimization problem, and propose several approaches for solving the optimization problem (leading to several scheduling policies). Finally, we describe our approach for predicting the future production of green energy, which is a needed input for solving the optimization problem.

### A. Green SLA Service

Each arriving job includes information about the number of VMs in the job, an estimate of its run-time, and a Green SLA, expressed as the percentage of energy used in executing the job that must be green. A job with a Green SLA of $x\%$ means that at least $x\%$ of the energy used to run the job must come from the Green Bus.

The provider charges clients a rate for each resource-time unit that can be run using either type of energy (e.g., \$0.24 per machine-hour). The provider charges a higher rate for each resource-time unit that should be run using only green energy (e.g., \$0.29 per machine-hour). The higher rate embodies the cost of providing the Green SLAs (e.g., the amortized capital cost of the solar plant and transfer switches) and a profit margin. The provider can reject a job at arrival if it does not have sufficient processing capacity and/or expected green energy production to execute the job and meet its Green SLA. However, if the job is admitted and the provider fails to meet the job's Green SLA, the provider must pay a penalty proportional to how much of the Green SLA was missed (e.g.,

$0.50 per machine-hour of Green SLA not fully powered by green energy).

### B. Scheduling Overview

The provider runs a scheduler that is designed to maximize profit while meeting jobs' Green SLAs, where the cost to the provider includes the cost for brown energy and any penalty that must be paid for violations of Green SLAs. Specifically, the scheduler divides time in a scheduling window (e.g., the next 48 hours) into epochs, with an epoch of time on a machine called a slot. It then predicts the green energy production, and produces a schedule that specifies for each time epoch in the scheduling window: (1) the number of VMs of each active job $j$ that should be running on each rack $r$, and (2) the power bus to which each rack should be connected. A slot in a rack connected to the Green Bus is called a green slot; one connected to the Mixed Bus is called a brown slot.

While the infrastructure described in Section III allows for arbitrary configurations of the power switches, the scheduler uses a more constrained configuration to simplify the scheduling problem. Specifically, the racks are divided into "left" and "right" partitions for each epoch (the vertical dash line in Figure 1 shows an example partitioning). The racks in the right partition will be completely powered with green energy during the epoch, while the racks in the left partition will be powered mostly with brown energy. A small amount of green power may be routed to the Mixed bus, but only if this excess green energy is insufficient to allow another entire rack to be moved to the green (right) partition. This excess green energy is used to reduce the amount of brown energy needed (and so reduces cost) but cannot be used toward meeting Green SLAs. In this approach, the leftmost rack will be the brownest rack (i.e., it has the highest brown-to-green energy ratio over time), the rightmost rack will be the greenest rack, and the racks become "greener" from left to right.

The scheduler runs at the beginning of an epoch to produce a new schedule when a new job arrives and/or when the system detects that Green SLAs may be violated because green energy production is lower than previously predicted.

To minimize energy consumption and cost, we turn servers off when they are not needed to run active jobs. For simplicity, we assume that data is stored on network-attached storage so that turning servers off does not affect data availability. We have solved this availability problem in a similar context [14].

VMs can be migrated live from one physical machine to another (i.e., a VM keeps running while it is being migrated). During migration, energy is consumed on both the source and destination machines.

### C. Optimization Framework

Table I lists the set of parameters for our optimization framework. Using these parameters, we formulate the optimization problem shown in Figure 2.

In this formulation, the profit (Equation 1) that the cloud provider earns is the sum of revenues earned from running admitted jobs minus the penalty incurred for missing any Green SLA and the energy cost of running the admitted jobs. An opportunity cost ($OppCost_j$), representing expected future earning that is given up when slots are assigned to current jobs, is also subtracted. We introduce this cost to ensure that VMs are not scheduled on "greener" racks than needed to satisfy their Green SLAs. Otherwise, the system may need to migrate VMs (incurring migration costs) or reject new jobs (loosing opportunities to increase profit) when they arrive in the future. The profit formulation does not include the capital and operating costs of the solar setup because all scheduling policies we seek to compare embody these same costs.

| Symbol | Meaning |
|---|---|
| $T$ | The number of time epochs in the scheduling window |
| $J$ | Set of active jobs |
| $V_j$ | Number of VMs in job $j$ |
| $G_j$ | Number of (remaining) green slots required for job $j$ |
| $T_j$ | Number of (remaining) time epochs in $j$'s execution |
| $R_b$ | Revenue earned per execution of job in a slot using brown energy |
| $R_g$ | Revenue earned per execution of job in a slot using green energy |
| $L$ | Penalty per slot of Green SLA not met |
| $RA$ | Number of racks that implement the Green SLA service |
| $RC$ | Capacity of (number of machines in) each rack |
| $EV$ | Amount of energy consumed by a machine running a VM in an epoch |
| $EM$ | Amount of energy consumed by the migration of a VM |
| $PB$ | Energy price that the cloud provider pays per brown slot |
| $OP_r$ | Opportunity cost for using a slot in rack $r$ |
| $GP_t$ | Predicted green energy production in epoch $t$ |
| $e^b_{rt}$ | Amount of brown energy scheduled for rack $r$ during epoch $t$ |
| $e^g_{rt}$ | Amount of green energy scheduled for rack $r$ during epoch $t$ |
| $x_{jrt}$ | Number of VMs of job $j$ scheduled on rack $r$ in epoch $t$ |
| $m_{jrt}$ | Number of VMs migrating out of rack $r$ at the start of epoch $t$ |

TABLE I. FRAMEWORK PARAMETERS. TIME EPOCHS IN THE SCHEDULING WINDOW ARE NUMBERED FROM 1 TO $T$. RACKS ARE NUMBERED FROM 1 TO $RA$, WHERE RACK $i$ IS ADJACENT TO THE LEFT OF RACK $i+1$ (LEFT-TO-RIGHT NUMBERING OF RACKS SHOWN IN FIGURE 1).

The revenue earned per job (Equation 2) is the revenue earned for the promised number of green slots and for the number of brown slots. The penalty (Equation 3) is proportional to the number of promised green slots replaced by brown slots because the cloud provider did not produce enough green energy while the job was running. The opportunity cost of each job (Equation 7) is the sum of the opportunity cost of all slots assigned to the job during its run-time. The cost to the provider of running the admitted jobs includes the cost of supplying racks with brown energy (Equation 8).

The optimization problem involves many constraints, with the main ones listed and described briefly in Figure 3.

We use the optimization solver to instantiate, for each time epoch $t$ in the scheduling horizon, the number of VMs from each job $j$ to run on each rack $r$ ($x_{jrt}$), the amount of brown energy supplied to each rack ($e^b_{rt}$), and the amount of green energy supplied to each rack ($e^g_{rt}$). Each time the problem is solved, $T_j$ and $G_j$ of each job that has already been running have to be updated to reflect how long the job has already run, and how many green slots it has already been allocated.

Note that our framework does not include the explicit management of the battery shown in Figure 1. Although we could have included this management [15], we have chosen to exclude it for simplicity. As we propose only a small power "smoothing" battery, this omission has little impact (if any) on our results. We have also chosen simplicity (e.g., constant brown energy pricing, constant penalty per missed slot) whenever the added complexity would be unlikely to affect our findings significantly. Finally, the framework currently focuses solely on the servers, assuming that cooling and network-attached storage fully rely on green energy and

$$
\begin{aligned}
Profit &= \sum_{j \in J}(Revenue_j - Penalty_j - OppCost_j) - \sum_{t=1}^{T}\sum_{r=1}^{RA} Cost(r,t) && (1)\\
Revenue_j &= G_j \times R_g + ((T_j \times V_j) - G_j) \times R_b && (2)\\
Penalty_j &= \begin{cases} L \times (G_j - NumActGreenSlots_j) & \textit{if } G_j > NumActGreenSlots_j \\ 0 & \textit{otherwise} \end{cases} && (3)\\
NumActGreenSlots &= \sum_{r=1}^{RA}\sum_{t=1}^{T}(x_{jrt} \times RackIsGreen_{rt}) && (4)\\
RackIsGreen_{rt} &= \begin{cases} 1 & \textit{if } e^g_{rt} \geq \sum_{j \in J}(x_{jrt} \times EV + NumMigrates_{jrt} \times EM) \\ 0 & \textit{otherwise} \end{cases} && (5)\\
NumMigrates_{jrt} &= \begin{cases} x_{jrt} - x_{jr(t-1)} & \textit{if } x_{jrt} > x_{jr(t-1)} \\ 0 & \textit{otherwise} \end{cases} && (6)\\
OppCost_j &= \sum_{r=1}^{RA}\sum_{t=0}^{T}(x_{jrt} \times OP_r) && (7)\\
Cost(r,t) &= e^b_{rt} \times PB && (8)
\end{aligned}
$$

Fig. 2. Optimization framework.

$$\forall_{j \in J}\forall_{0<t\leq T_j}\sum_{r=1}^{RA} x_{jrt} = V_j \Rightarrow \text{Must run all VMs of each job } j \text{ in every epoch during } j\text{'s run-time} \quad (9)$$

$$\forall_{0<r\leq RA}\forall_{0<t\leq T}\sum_{j \in J} x_{jrt} \leq RC \Rightarrow \text{Total number of VMs running in a rack must not exceed its capacity} \quad (10)$$

$$\forall_{0<r\leq RA}\forall_{0<t\leq T}(((EV \times \sum_{j \in J} x_{jrt}) + (EM \times \sum_{j \in J} NumMigrates_{jrt})) \leq (e^b_{rt} + e^g_{rt})) \Rightarrow \quad (11)$$
$$\text{Enough energy must be scheduled for each rack to run VMs scheduled there and migrate VMs moving to another rack}$$

$$\forall_{0<t\leq T}\ \nexists_r|(RackIsGreen_{(r-1)t} = 1) \wedge (RackIsGreen_{rt} = 0) \wedge (RackIsGreen_{(r+1)t} = 1) \Rightarrow \quad (12)$$
$$\text{All green racks must be next to each other}$$
$$\forall_{0<t\leq T}\ \nexists_r|(r \neq RA) \wedge (RackIsGreen_{rt} = 1) \wedge (RackIsGreen_{RAt} = 0) \Rightarrow \text{Green racks must start from right end}$$

$$\forall_{0<r\leq RA}\forall_{0<t\leq T}\sum_{r=1}^{RA} e^g_{rt} \leq GP_t \Rightarrow \text{Can schedule no more green energy than the amount of expected production} \quad (13)$$

Fig. 3. Optimization constraints.

(larger) batteries. We will eliminate this assumption in our future work.

### *D. Solving the Optimization Problem*

**Simulated Annealing.** We use Simulated Annealing (SA) [29] to solve the above non-linear formulation. Recall that this optimization is executed each time a new job arrives, or the system detects that one or more Green SLAs might be missed.

When a new job arrives, SA first solves a simpler, linear reformulation of the problem (see below) to schedule the new job, assuming that the current schedule for jobs already in the system cannot be changed. This produces a starting point for SA. When the system detects that Green SLAs might be violated, the current schedule is the starting point.

Starting at this point, SA iteratively explores new schedules, relying on randomization to avoid local minima. To limit the size of the search space, SA does not explore completely random new schedules. Instead, it maintains three job sets: (1) an "unmet" set of jobs whose Green SLAs are expected to be violated under the current schedule, (2) an "extra" set of jobs that are expected to receive more green energy than is needed to satisfy their Green SLAs, and (3) a "migration" set of jobs whose schedules contain more than a threshold of migrations in the future. SA produces a new schedule by randomly choosing a small subset of each job set, removing them from the current schedule, and rescheduling them as if they were new arrivals using the linear reformulation.

For a new job, if SA cannot find a schedule that increases profit, the job is rejected. Otherwise, the best schedule found by SA is adopted. In the case of detecting possible Green SLA violations, the best schedule found by SA is adopted, regardless of whether one or more Green SLAs are still expected to be missed or not. Once admitted, jobs must be executed to completion. If Green SLAs are missed, the cloud provider must pay the penalty.

**Linear Programming.** We can reformulate the optimization problem in Figure 2 into a simpler one that is solvable using Linear Programming (LP). In this reformulation, the optimization problem becomes one of minimizing the sum of the penalty, opportunity cost, and migration cost. We assume that the racks in the green partition are always completely full, allowing the green/brown energy routing to be predetermined into the future. Intuitively, the minimization problem leads to a solution that tries to meet Green SLAs (minimizes penalty) without unnecessarily using greener slots than is needed (minimizes opportunity cost) and without unnecessarily moving VMs (minimizes migration cost).

The reformulated problem is actually a Mixed Integer Linear Programming (MILP) problem that we solve using a MILP solver. Since MILP solvers are only efficient for reasonably sized problems, LP can only be used to schedule a small number of jobs. Thus, when using just LP, we assume that once a job has been scheduled, its schedule cannot be changed in the future. When a new job arrives, it is scheduled using only resources (i.e., slots) that are not already used in the current schedule. It is rejected if the solution leads to lower profits. As already described, when used as a part of SA, LP may be used to produce a schedule for a small set of jobs, assuming the current schedule for the remaining jobs is fixed.

**Greedy Heuristics.** As baselines for comparison, we propose two greedy heuristic placement schemes. The first is a First-Fit (FF) scheme that is oblivious to energy type and admits all jobs as long as there is sufficient computing capacity. The second is a more sophisticated, green-energy-aware heuristic that we call Static Green-aware Placement (SGP). Specifically, when a job arrives with a Green SLA of $g\%$, SGP places the job's VMs on the rack $r$ with expected green percentage during the job's runtime closest to but not less than $g\%$. If there are more VMs than available slots in $r$, the needed percentage of green is recomputed for the excess VMs, and the same placement procedure is repeated. VMs are never migrated. The job is rejected if a placement that is expected to meet or exceed the job's Green SLA cannot be found.

### *E. Green Energy Prediction*

We use the method from [14] to predict solar energy production. This method combines the model from [30] with the approach for improving accuracy from [13]. Specifically, the model relates solar energy generation to cloud cover as $E_p(t) = B(t)(1 - CloudCover)$, where $E_p(t)$ is the amount of energy predicted for time $t$, $B(t)$ is the amount of energy expected under ideal sunny conditions for time $t$, and $CloudCover$ is the forecasted percentage cloud cover. For the cloud cover information, we use forecasts from Intellicast.com, which predicts $CloudCover$ for each hour of the next 48 hours. This leads to prediction granularity (i.e., $t$) of one hour. We set $B(t)$ for each hour of the day to the amount of energy generated during that hour on the day with the highest energy generation from the previous month.

Of course, weather forecasts are sometimes wrong, which may lead to inaccurate predictions. To improve accuracy, we compute $CloudCover$ from the amount of energy generated in the previous hour. This approach then compares the accuracy of the two methods, and uses the most accurate one to predict the remainder of the horizon. For example, at the beginning of hour $t$, we compute $CloudCover$ for the next hour using (1) the weather forecast, and (2) the energy produced in hour $t-1$. At the beginning of hour $t+1$, we compare the accuracy of the two methods and use the best one to predict the remainder of the horizon. At hour $t+2$, the process repeats.

## V. Evaluation

### *A. Methodology*

We use simulation to evaluate our framework and policies. Our simulator takes a workload trace, a solar energy production trace, a solar energy prediction trace, and the price for brown energy as inputs. Using these inputs, it simulates job arrivals and executions using a given scheduling policy, while tracking job energy consumption and whether Green SLAs are met.

**Datacenter.** We simulate a datacenter containing 16 racks, each rack containing 40 servers. Each server consumes 140W when executing a VM, giving a peak datacenter power demand of 89.6kW. The 140W value is the measured power consumption of a server equipped with a 2.4GHz 4-core Xeon CPU, 8GB of memory, 1 7200rpm disk, and a 1Gb Ethernet card.

**Workload.** We use a trace from the Parallel Workload Archive [31]. The trace is an 8-month long log of Intrepid, a 40-rack Blue Gene/P system deployed at Argonne National Laboratory. We selected an arbitrary 48-hour portion of the trace. To make simulation time reasonable, we scale the trace down to fit within our smaller datacenter. Specifically, we scale down each job's node demand by a factor of 64, reducing the total system size from 40960 to 640 nodes. We correspondingly scale up job run-times by a factor of 8 (assuming that jobs run longer when running on smaller numbers of nodes).

The chosen 48-hour period contains 480 jobs. We filtered out 15 very large jobs that would have been rejected by all scheduling policies because of capacity constraints. We also filtered out 24 jobs that ran for longer than 48 hours since we currently only predict green energy production for 48 hours into the future. We do this latter filtering just for simplicity; it is possible to modify our green energy prediction method and/or scheduling policies to handle long running jobs. The final workload trace contains 441 jobs, with a peak processing demand of 627 nodes. We map this workload into our environment by assuming that each job can be split into VMs, one VM per node requested by the job.

As previously discussed, each arriving job specifies the number of VMs needed, an estimated run-time (which we assume to be accurate in our simulation), and a Green SLA. The Green SLA of each job is chosen randomly according to a uniform distribution from the set $0\%, 25\%, 50\%, 75\%, 100\%$. Figure 4 plots the workload power demand over time assuming that all jobs are admitted.

**Energy Prices.** We use the average brown energy price of our state, 10.55 cents/kWh [32]. We assume self-generation of solar energy, so that green energy has zero (incremental) cost. Note, however, that our framework can be easily extended to account for a non-zero green energy cost.

**Service Pricing.** Customers pay \$0.29 per contracted VM-hour in their Green SLAs and a fee of \$0.24 per each of the remaining contracted VM-hours. These prices are modeled

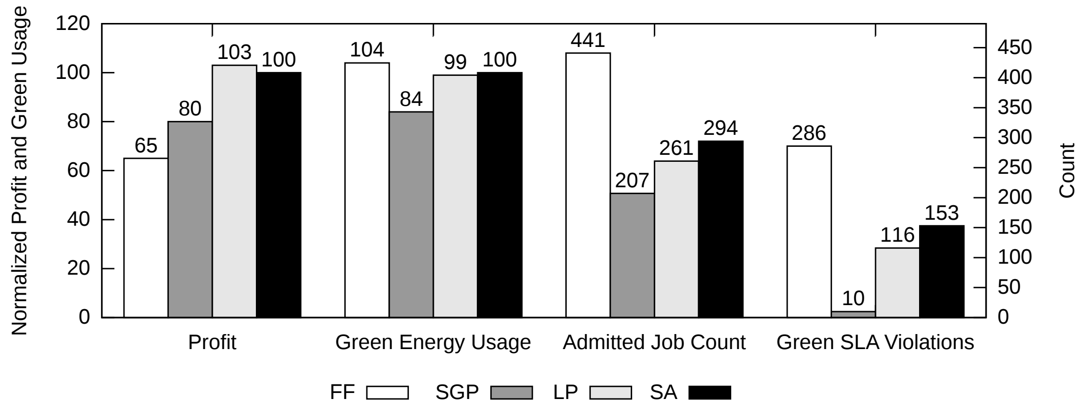



Fig. 5. Comparison of normalized profit, normalized green energy use, number of admitted jobs, and number of Green SLA violations for Medium days.

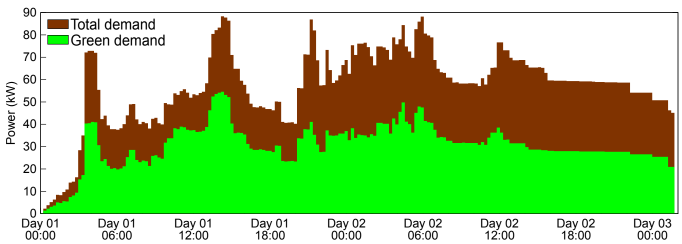



Fig. 4. Power demand of workload, assuming all jobs are admitted.

after GreenQloud's pricing [33] and Amazon EC2's pricing [34], respectively, for a large VM instance running Linux. These prices place a premium of approximately 20% for each contracted VM-hour in a Green SLA.

If the cloud provider fails to meet a job's Green SLA, it pays a penalty of \$0.50 per missed VM-hour. We have chosen a relatively large penalty (10 times the 5 cents premium for requiring green energy) to ensure that the cloud provider does not lightly dismiss Green SLAs to garner greater profits. We explore the sensitivity of our policies to different ratios between the service prices and penalty below.

With the above prices and penalty, if the provider admits a 1-VM job that runs for 1 hour and has a Green SLA of 100%, it earns 29 cents if it meets the job's Green SLA, earns 4 cents (29 cents fee - 25 cents penalty) if it misses the Green SLA by 50%, and pays the client 21 cents (29 cents fee - 50 cents penalty) if it does not use green energy to run the VM at all.

**Opportunity Cost.** We could compute the opportunity cost by scheduling a representative workload over an appropriate time frame, and computing the average profit earned per slot within each rack. However, for simplicity, we currently use an $\epsilon$ value that increases from the brownest rack to the greenest rack. This $\epsilon$ opportunity cost is sufficient to prevent unnecessary placement of VMs on green slots, which may later require migration so that the slots can be recovered to admit new jobs.

**Solar Power Generation.** We model the solar power generation as a scaled-down version of a Rutgers's solar farm, which has a rated production capacity of 1.4MW. We scale the farm's production down to 400 solar panels capable of producing 92kW. After derating, this system's peak production can provide 75% of the peak power consumption of our simulated datacenter. We choose 75% because on days with high solar energy production, this is sufficient to admit almost the entire workload while solar energy is being produced.

We evaluate our scheduling policies using three pairs of consecutive days with different amounts of solar energy production. Specifically, we select two sunny days (5/9/11 and 5/10/11) with high solar energy production (totaling 1.23MWh), two days (6/16/11 and 6/17/11) with medium solar energy production (totaling 740kWh), and two days (5/15/11 and 5/16/11) with low solar energy production (totaling 240kWh). We call these the "High", "Medium", and "Low" days, respectively.

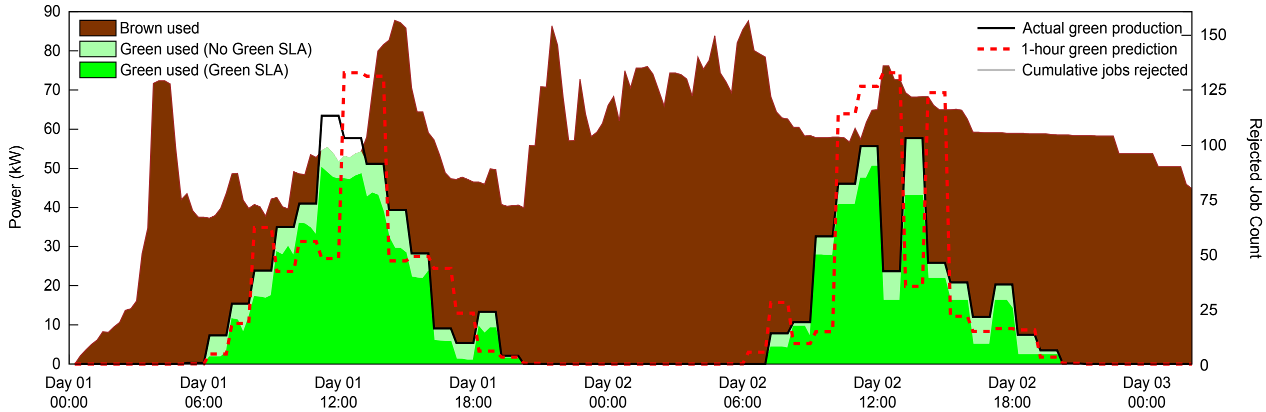
Interestingly, the three pairs of days show different levels of accuracy for predictions of solar energy production. Predictions are mostly accurate for the High days, although there are some under-predictions for the 1st day. Predictions are also mostly accurate for the Low days. However, predictions for the Medium days contain significant errors, which are mostly over-predictions. We evaluated our solar energy predictions in more detail in [14].

**Optimization.** The scheduling horizon is set to 48 hours, the extent of time into the future that we currently predict green energy production. The scheduling horizon is divided into 15-minute time epochs, implying that the scheduling policy is rerun at most once every 15 minutes. We use the Gurobi solver [35] to solve the MILP optimization problem described in Section IV.

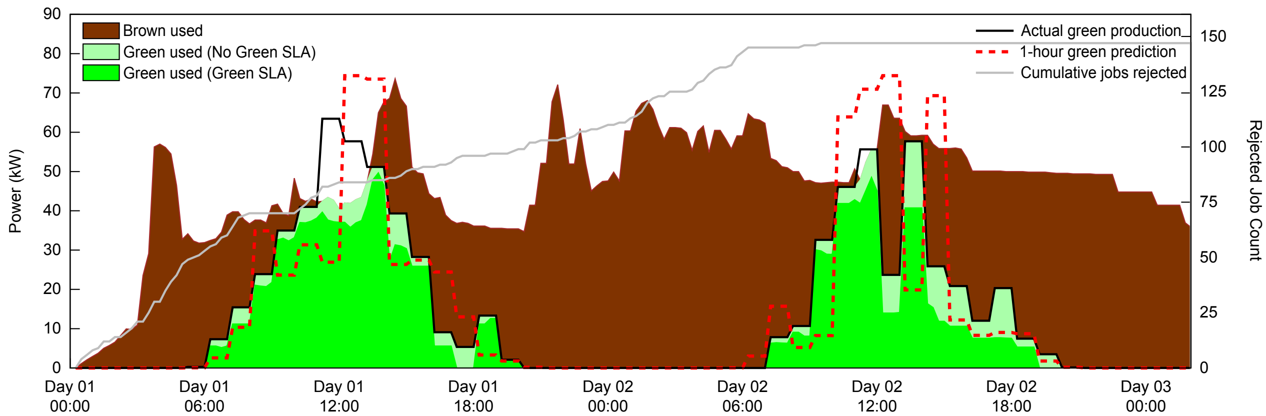
### B. Results

Figure 5 compares the profit, green energy usage, number of admitted jobs, and number of missed Green SLAs achieved on the Medium days by the four policies described in Section IV-D. The profit and green energy usage results are normalized to those of SA (Y-axis on the left), whereas the numbers of admitted jobs and Green SLA violations are absolute (Y-axis on the right). These results show that the two optimization-based policies, SA and LP, can significantly outperform the two greedy policies, with FF and SGP achieving only 65% and 80%, respectively, of SA's profit.

Interestingly, the greedy policies under-perform the optimization-based policies for opposing reasons. FF, as expected, is overly aggressive since it admits all jobs, regardless of their Green SLAs and expected green energy production. Thus, it violates many Green SLAs and incurs large penalties.




(a) FF




(b) SA

Fig. 6. Power profile of FF and SA for the Medium days. The "Green used (Green SLA)" areas represent green energy used for meeting the Green SLAs of admitted jobs. The "Green used (No Green SLA)" areas represent green energy used to run jobs beyond that required to meet Green SLAs. The "Brown used" areas represent brown energy used. The dashed (red) lines show the 1-hour ahead predictions of green energy production. The solid (black) lines show the actual green energy production. The solid (grey) line in (b) shows the number of jobs rejected (Y-axis on the right). Note that FF does not reject any jobs.

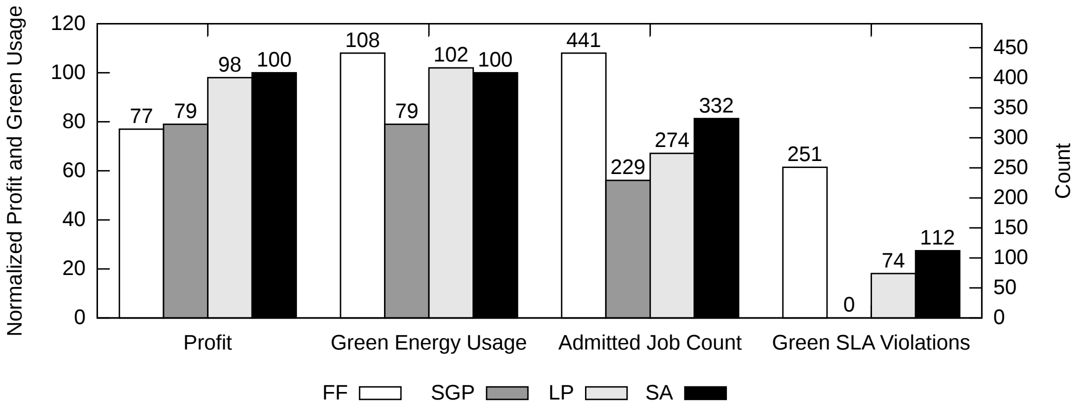
However, it uses more green energy because the other policies sometimes reject jobs even when green energy is available.

In contrast, SGP is overly conservative in trying to not violate any Green SLA at all. Thus, it misses opportunities where the cloud provider can earn a profit because the penalty is relatively small compared to the revenue earned (e.g., when Green SLAs are only missed by small amounts). However, SGP violates the fewest Green SLAs among all the policies; it only violates the Green SLAs of 10 out of 207 admitted jobs (5%). This can be advantageous if violating Green SLAs has negative implications beyond the penalty; e.g., clients seeking to reduce the carbon footprint of their computing workloads may become unhappy if too many of the Green SLAs are missed, despite the compensating penalties.

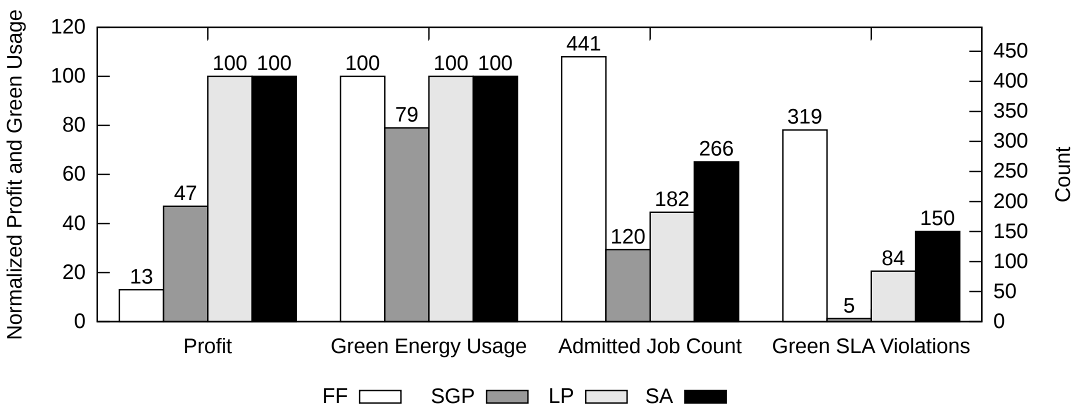
For the current parameters, there is little difference between the performance of LP and SA for maximizing profit. SA admits jobs more aggressively, but violates Green SLAs more frequently, and so pays higher penalties. LP outperforms SA slightly (3% higher profit) for the Medium days because the predicted green energy production is higher than the actual production. Thus, SA's aggressive admission of jobs becomes a disadvantage, causing it to incur higher penalties than predicted, and so lower profits. SA also takes substantially longer (1.4 secs on average in our experiments with a 2.4GHz Xeon server) to compute a schedule than LP (0.1 secs on average). However, both overheads are low compared to our relatively long epochs (15 minutes). Also, LP runs less frequently (only when a new job arrives, i.e., 441 times) than SA (632 times) in our experiments. We explore the scalability of SA's and LP's runtimes with datacenter and workload sizes below.

For a more detailed look at the behaviors of the policies, Figures 6(a) and 6(b) show the power consumption profiles of FF and SA for the Medium days. Overall, SA consumes less energy because it rejects jobs when it predicts that there will not be sufficient green energy to profitably admit the jobs. At peak green energy production, especially if green production is higher than predicted, this may cause some green energy to go unused; of course, this green energy can be either net-metered or stored in batteries for later use. Both FF and SA (LP as well) are successful at using the vast majority of the green energy for meeting Green SLAs. However, when more green energy is produced than predicted (e.g., after 12noon on the second day), a larger fraction of the green energy may be used to power slots that could have been powered by brown energy without violating Green SLAs.

**Impact of ratio between green revenue and penalty.** The ratio between the fee for meeting Green SLAs (green revenue) and the penalty for not meeting them significantly impacts the behaviors of LP and SA, and their performance relative to the greedy policies. As the penalty increases compared to the revenue, both LP and SA will admit fewer jobs to avoid violations of Green SLAs. Thus, their performance will




(a) High Days




(b) Low Days

Fig. 7. Comparison of normalized profit, normalized green energy use, number of admitted jobs, and number of green SLA violations for High and Low days.

becomes comparable to that of SGP. For example, on High days, the ratio of SGP's profit to SA's profit is 0.6:1, 0.79:1, and 0.85:1 for low (\$0.20), medium (\$0.50), and high (\$0.80) penalty, respectively. We observe similar results for Medium and Low days. Further, FF's relative performance will worsen because its obliviousness to green energy production becomes increasingly expensive. As the penalty decreases compared to the revenue, both LP and SA will admit more jobs to increase profit, even though more Green SLAs will be missed. Thus, their performance will become more comparable to that of FF. SGP's relative performance will worsen because it does not recognize that violations of Green SLAs can be profitable.

**Impact of green energy availability.** We now evaluate the policies across the three pairs of days with different levels of green energy production. Figures 7(a) and 7(b) show the results for the High and Low days. As one might expect, the advantages of the optimization-based policies become more clear when the availability of green energy is more limited. On the Low days, FF and SGP only achieve 13% and 47% of SA's profit. On the other hand, as green energy becomes more available, the performance of the policies becomes more comparable. On the High days, FF achieves 77% of SA's profit. Interestingly, SGP performs similarly for Medium and High days when compared to SA (80% and 79% of SA's profits, respectively). Again, this is because SGP is overly conservative, and so it misses opportunities for increasing profit (while also missing a small number of Green SLAs).

SA outperforms LP slightly on the High days because the predicted green energy production is lower than the actual production. In this case, SA can migrate VMs from jobs receiving more green energy than expected to browner racks, allowing it to admit more jobs.

We have also simulated scenarios with peak green energy production providing 25% and 50% (compared to the above base case of 75%) of the peak power consumption of the datacenter. Trends in these results are consistent: the performance gap between the optimization-based and heuristic-based policies widens as the availability of green energy decreases.

**Impact of green energy prediction inaccuracies.** Inaccurate green energy predictions can harm profit. Over-predictions (i.e., predicted production is greater than actual production) can be especially bad since they also lead to increased violations of Green SLAs. Thus, it may be desirable to make the green energy predictions more conservative.

To study the impact of conservative predictions, we rerun our simulations while reducing all $B(t)$ (Section IV-E) values by 10%, 20%, and 30%. These conservative predictions lead to slightly lower profits for LP ($<$7%) but significantly fewer missed Green SLAs (in one case, over 30% fewer). On the High and Low days, these conservative predictions also lead to slightly lower profits for SA ($<$6%) but significantly fewer missed Green SLAs (as many as 20% fewer). For the Medium days, SA's profit actually increases (because the original predictions were over-predictions) by 5% and the number of missed Green SLAs decreased by 21%. In fact, SA now outperforms LP, even when LP is using the original green energy predictions.

Thus, it is worthwhile to make the green energy predictions more conservative than the method described in Section IV-E, perhaps by up to 30%.

**Scalability of SA and LP.** We explore the scalability of SA and LP by running experiments for datacenters with 4, 8, and 32 racks. We also scale the workload proportionally (e.g., the workload of a datacenter with 32 racks is twice that of a datacenter with 16 racks). As we are increasing the problem size exponentially, the runtimes also grow exponentially. However, the overheads for a 32-rack system are still low compared to our 15-minute epochs. Average runtimes for LP are 0.01, 0.05, 0.1, 0.7 secs for 4, 8, 16, and 32 racks, respectively. Average runtimes for SA are 0.07, 0.26, 1.4, and 11.8 secs. As systems and workloads increase further, the provider may consider utilizing more machines to run our policies, or simply increase the epoch length.

## VI. Conclusion

In this paper, we proposed that HPC cloud service providers should offer a new class of green service. In this service, each client job specifies a Green SLA, which is the minimum percentage of green energy that must be used to run the job. We then proposed a power distribution and control infrastructure, together with an optimization-based scheduling framework and policies, for providing such a Green SLA service. We also proposed two simple greedy heuristic policies for achieving the same goals. We evaluated our proposals extensively using simulations. Our evaluation results showed that the optimization-based policies can significantly outperform the greedy policies. The results also showed that the choice of optimization-based policy depends on whether the cloud provider prefers to accept more jobs or violate fewer Green SLAs. We conclude that a Green SLA service that uses our policies can be useful for the provider to attract environmentally conscious clients, and especially those who require strict guarantees on their use of green energy.

## References


[1] J. Mankoff, R. Kravets, and E. Blevis, "Some Computer Science Issues in Creating a Sustainable World," *Computer*, vol. 41, no. 8, 2008.

[2] Global Reporting Initiative, https://www.globalreporting.org.

[3] Green House Data, http://www.greenhousedata.com.

[4] AISO.net, http://www.aiso.net.

[5] GreenQloud, http://greenqloud.com.

[6] Green House Data, http://www.greenhousedata.com/about-us/green-data-center/, Retrived in February 2013.

[7] UK Government, "Carbon Reduction Commitment," http://www.carbonreductioncommitment.info/.

[8] Apple Inc., "Apple and the Environment," http://www.apple.com/-environment/renewable-energy/, Retrived in February 2013.

[9] R. Miller, "Data Centers Scale Up Their Solar Power," May 2012.

[10] EcobusinessLinks, "Green Hosting - Sustainable Solar & Wind Energy Web Hosting," http://www.ecobusinesslinks.com/green_webhosts/, Retrieved in February 2013.

[11] B. Aksanli, J. Venkatesh, L. Zhang, and T. Rosing, "Utilizing Green Energy Prediction to Schedule Mixed Batch and Service Jobs in Data Centers," in *Proceedings of the 4th Workshop on Power-Aware Computing and Systems*, October 2011.

[12] N. Deng, C. Stewart, and J. Li, "Concentrating Renewable Energy in Grid-Tied Datacenters," in *Proceedings of the International Symposium on Sustainable Systems and Technology*, May 2011.

[13] I. Goiri, K. Le, M. E. Haque, R. Beaucheau, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "GreenSlot: Scheduling Energy Consumption in Green Datacenters," in *International Conference for High Performance Computing, Networking, Storage and Analysis*, November 2011.

[14] I. Goiri, K. Le, , T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "GreenHadoop: Leveraging Green Energy in Data-Processing Frameworks," in *ACM European Conference on Computer Systems*, April 2012.

[15] I. Goiri, W. Katsak, K. Le, T. D. Nguyen, and R. Bianchini, "Parasol and GreenSwitch: Managing Datacenters Powered by Renewable Energy," in *International Conference on Architectural Support for Programming Languages and Operating Systems*, March 2013.

[16] A. Krioukov, C. Goebel, S. Alspaugh, Y. Chen, D. Culler, and R. Katz, "Integrating Renewable Energy Using Data Analytics Systems: Challenges and Opportunities," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, March 2011.

[17] A. Krioukov, S. Alspaugh, P. Mohan, S. Dawson-Haggerty, D. E. Culler, and R. H. Katz, "Design and Evaluation of an Energy Agile Computing Cluster," University of California at Berkeley, Tech. Rep. EECS-2012-13, January 2012.

[18] K. Le, R. Bianchini, M. Martonosi, and T. D. Nguyen, "Cost- And Energy-Aware Load Distribution Across Data Centers," in *Workshop on Power Aware Computing and Systems*, October 2009.

[19] C. Li, A. Qouneh, and T. Li, "iSwitch: Coordinating and Optimizing Renewable Energy Powered Server Clusters," in *International Symposium on Computer Architecture*, June 2012.

[20] 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 *International Conference on Measurement and Modeling of Computer Systems*, June 2012.

[21] N. Sharma, S. Barker, D. Irwin, and P. Shenoy, "Blink: Managing Server Clusters on Intermittent Power," in *International Conference on Architectural Support for Programming Languages and Operating Systems*, March 2011.

[22] C. Stewart and K. Shen, "Some Joules Are More Precious Than Others: Managing Renewable Energy in the Datacenter," in *Workshop on Power Aware Computing and Systems*, October 2009.

[23] D. Breitgand and A. Epstein, "SLA-Aware Placement of Multi-Virtual Machine Elastic Services in Compute Clouds," in *International Symposium on Integrated Network Management*, May 2011.

[24] I. Goiri, F. Julià, R. Nou, J. L. Berral, J. Guitart, and J. Torres, "Energy-Aware Scheduling in Virtualized Datacenters," in *IEEE International Conference on Cluster Computing*, September 2010.

[25] K. Le, J. Zhang, J. Meng, Y. Jaluria, T. D. Nguyen, and R. Bianchini, "Reducing Electricity Cost Through Virtual Machine Placement in High Performance Computing Clouds," in *International Conference for High Performance Computing, Networking, Storage and Analysis*, November 2011.

[26] A. Verma, P. Ahuja, and A. Neogi, "pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems," in *International Middleware Conference*, December 2008.

[27] S. Pelley, D. Meisner, P. Zandevakili, T. F. Wenisch, and J. Underwood, "Power Routing: Dynamic Power Provisioning in the Data Center," in *International Conference on Architectural Support for Programming Languages and Operating Systems*, March 2010.

[28] Maxim Integrated, "Switching Between Battery and External Power Sources," http://www.maximintegrated.com/app-notes/index.mvp/id/1136, Retrieved in February 2013.

[29] S. Kirkpatrick, "Optimization by Simulated Annealing: Quantitative Studies," *Journal of Statistical Physics*, vol. 34, no. 5, 1984.

[30] N. Sharma, J. Gummeson, D. Irwin, and P. Shenoy, "Cloudy Computing: Leveraging Weather Forecasts in Energy Harvesting Sensor Systems," in *International Conference on Sensor Mesh and Ad Hoc Communications and Networks*, June 2010.

[31] D. Feitelson, "Parallel Workload Archive," http://www.cs.huji.ac.il/-labs/parallel/workload/l_anl_int/index.html.

[32] Energy Information Administration, "Average Retail Price of Electricity to Ultimate Customers by End-Use Sector, by State," http://www.eia.gov/electricity/monthly/epm_table_grapher.cfm?t=epmt_5_6_b, Retrieved in January 2013.

[33] GreenQloud, http://greenqloud.com/pricing/, Retrieved in February 2013.

[34] Amazon, http://aws.amazon.com/pricing/ec2/, Retrieved in February 2013.

[35] Gurobi Optimization Inc., "Gurobi Optimization," http://www.gurobi.com.