Parasol and GreenSwitch: Managing Datacenters Powered by Renewable Energy *

Íñigo Goiri, William Katsak, Kien Le†, Thu D. Nguyen, and Ricardo Bianchini

Department of Computer Science
Rutgers University
{goiri,wkatsak,lekien,tdnguyen,ricardob}@cs.rutgers.edu

Abstract

Several companies have recently announced plans to build "green" datacenters, i.e. datacenters partially or completely powered by renewable energy. These datacenters will either generate their own renewable energy or draw it directly from an existing nearby plant. Besides reducing carbon footprints, renewable energy can potentially reduce energy costs, reduce peak power costs, or both. However, certain renewable fuels are intermittent, which requires approaches for tackling the energy supply variability. One approach is to use batteries and/or the electrical grid as a backup for the renewable energy. It may also be possible to adapt the workload to match the renewable energy supply. For highest benefits, green datacenter operators must intelligently manage their workloads and the sources of energy at their disposal.

In this paper, we first discuss the tradeoffs involved in building green datacenters today and in the future. Second, we present Parasol, a prototype green datacenter that we have built as a research platform. Parasol comprises a small container, a set of solar panels, a battery bank, and a grid-tie. Third, we describe GreenSwitch, our model-based approach for dynamically scheduling the workload and selecting the source of energy to use. Our real experiments with Parasol, GreenSwitch, and MapReduce workloads demonstrate that intelligent workload and energy source management can produce significant cost reductions. Our results also isolate the cost implications of peak power management, storing energy on the grid, and the ability to delay the MapReduce jobs. Finally, our results demonstrate that careful workload and energy source management can minimize the negative impact of electrical grid outages.

Categories and Subject Descriptors C.m [*Computer Systems Organization*]: Miscellaneous; D.4.1 [*Operating Systems*]: Process Management—Scheduling

Keywords Renewable energy, batteries, scheduling, datacenters.

* This work was partially supported by NSF grant CSR-1117368, and the Rutgers Green Computing Initiative. †Kien Le is now with A10 Networks.

ASPLOS'13, March 16–20, 2013, Houston, Texas, USA.

1. Introduction

Datacenters range from small sets of servers in machine rooms to several thousands of servers housed in warehouse-size installations [23, 44]. One may find the former datacenters at universities and enterprises, and the latter at Internet service companies like Google and Microsoft. Collectively, datacenters consume a massive amount of energy. Estimates for 2010 indicate that datacenters consume around 1.5% of the total electricity used world-wide [23]. This translates into high carbon emissions as most of this energy 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 [31].

Given these emissions and increasing societal awareness of climate change, governmental agencies, non-profits, and the public at large are starting to demand cleaner products and services. As a result, several companies have announced plans to build "green" datacenters, i.e. datacenters partially or completely powered by renewable energy such as solar or wind energy. These datacenters will either generate their own renewable energy (self-generation) or draw it directly from an existing nearby plant (co-location). For example, Apple is building a 20MW solar array for its North Carolina datacenter [6]. McGraw-Hill has recently completed a 14MW solar array for its datacenter [7]. Green House Data and AISO are two small cloud service providers that operate entirely on renewable energy [1, 15]. Other examples can be found in [8].

We argue that it is irrelevant whether these companies are investing in co-located or self-generated renewable energy for market positioning, public relations, cost, or environmental reasons. The fact is that they are expecting benefits from these investments. For example, companies may be able to attract customers who value their investments in renewables. Moreover, despite their decreasing-but-still-high capital costs, exploiting solar and/or wind energy in datacenters may reduce overall energy costs [11], peak grid power costs [34], or both [12]. We expect that an increasing number of companies will see benefits in exploiting renewables.

However, we do *not* argue that co-location or self-generation is the best approach for all datacenter operators. For example, Google prefers to invest in renewables by financing new plants and pumping the produced energy into the electrical grid. Rather, we argue that co-location or self-generation will be the approach of choice for many operators, as suggested by [1, 6–8, 15].

Co-location and self-generation are interesting from a research perspective as well: solar and wind energy are intermittent, which requires approaches for tackling the energy supply variability. One approach is to use batteries and/or the electrical grid as a backup for the renewable energy. It may also be possible to adapt the workload

(the energy demand) to match the renewable energy supply [2, 11, 12, 24, 25]. For highest benefits, green datacenter operators must intelligently manage their workloads and the sources of energy at their disposal. For example, when the workload is deferrable (i.e., it can be delayed within a time bound), it may be appropriate to delay some of the load, and store the freed-up renewable energy in the batteries for later use (e.g., to shave an expected load peak when the renewable energy is not available). As far as we know, current green datacenter operators do not manage their energy sources and workloads in this manner.

We set out to build software and hardware to explore these issues. This paper describes some of our main efforts. First, we collect data from a large number of sources to quantify the tradeoffs involved in building solar- and/or wind-powered datacenters today and in the future. We pay particular attention to the evolving space requirements and capital costs of these technologies. We speculate that today's space requirements and capital costs may be cut in half in the 2020-2030 time frame.

Second, we present Parasol, a solar-powered micro-datacenter that we have built as a research platform. Parasol comprises a small container, a set of solar photovoltaic (PV) panels, batteries, and a grid-tie. The container houses two racks of energy-efficient servers and networking equipment. Parasol uses air-side economizer cooling (or simply "free cooling") whenever outside temperatures are low enough, and regular air conditioning otherwise.

Third, we describe GreenSwitch, our framework and system for scheduling workloads and selecting the source of energy to use (solar, battery, and/or grid) at each point in time. GreenSwitch is based on (1) predictions of future renewable energy availability; (2) predictions of future computational load; (3) the current amount of energy stored in the batteries; (4) analytical models of workload behavior, battery use, and electricity cost; and (5) the characteristics of the green datacenter and the prices of the grid energy and peak power. GreenSwitch seeks to minimize the overall cost of electricity, while respecting the characteristics of the workload and battery lifetime constraints. GreenSwitch can also be used to manage workloads and energy sources during electrical grid outages.

Fourth, we perform a large number of real day-long experiments with Parasol and MapReduce (Hadoop) workloads. We consider both deferrable and non-deferrable workloads. Our results demonstrate that smart workload and energy source management can indeed produce significant reductions in cost and carbon footprint. In contrast to prior work on leveraging batteries, e.g. [14], we find that batteries may not be as cost-effective when the system also includes a solar array and the ability to store energy on the grid ("net metering"). Our results also isolate the impact of peak power management, net metering, and the ability to delay the MapReduce jobs. Finally, our results demonstrate that careful workload and energy source management can minimize the negative impact of grid outages on the overall performance of a workload.

Through GreenSwitch, Parasol is the first green datacenter prototype to dynamically manage workload demands, multiple energy sources (renewable energy, batteries, and grid), and multiple energy stores (batteries and net metering), all at the same time. Throughout the paper, we mention important effects that we would have missed if we had used simulations or a simpler prototype in our study.

We conclude that green datacenters represent an increasingly interesting topic from many perspectives. We hope that Parasol and GreenSwitch will entice other researchers to consider these datacenters. In fact, we plan to enable other research groups to use Parasol from their remote sites through virtualization.

2. Related Work

Viability of green datacenters. Many recent papers have focused on datacenters that generate their own renewable energy [2, 3, 10–12, 24–26, 28, 29, 37, 39]. This paper adds to this body of work.

Ren *et al.* evaluated various ways for datacenters to incorporate renewable energy: self-generation, off-site generation feeding into the grid, power purchase agreements feeding into the grid, and renewable energy certificates [34]. They found that *both self-generation and off-site generation can lower costs*, as well as lower carbon footprints. In addition, they found self-generation to be the best approach when carbon reduction targets are moderate (up to 30%), especially due to its ability to reduce peak grid power cost.

Unfortunately, Ren *et al.* considered a low and constant grid energy price ($0.05/kWh), which made self-generation relatively expensive in comparison. A major benefit of self-generation is its ability to lower energy costs significantly under the pervasive off-peak/on-peak energy pricing. Had Ren *et al.* considered this type of pricing, and energy prices that are more representative of North America and Europe (e.g., $0.07-$0.14/kWh in the US [43]), they would have seen a larger range of scenarios where self-generation is cheaper than other approaches.

In contrast with this prior study, our analysis of viability focuses on the space requirements and capital costs of self-generation with wind and solar, now and in the future.

In a well-known blog post [17], James Hamilton from Amazon.com criticized Apple's 20MW solar array as consuming too much space for the amount of energy it will provide.

We agree with Hamilton's observations; solar energy currently consumes a significant amount of space and is still expensive. However, we take a broader perspective. First, as we illustrate in this paper, these space requirements and costs will continue to decrease quickly in the future. Second, many datacenters do not consume anywhere near maximum power most of the time. Third, many datacenter operators may want to draw only a moderate fraction of the energy consumed by their datacenters from renewables. Fourth, wind requires less space and lower capital expenditures than solar, and may provide a better alternative. (Although Parasol is solar-powered, our approaches for managing workloads and energy sources applies to any other technology that produces variability. Obviously, our approaches work best when this variability can be accurately predicted.) Finally, even when operators decide to leverage renewable energy for reasons other than electricity costs or the environment, the bottom line is that their workloads and energy sources must be managed properly for maximum benefits. Understanding how to do this management is a key goal of this paper.

Real implementations of green datacenters. Researchers from UMass at Amherst have built Blink [37], a cluster of 10 laptop motherboards powered by two micro wind turbines and two solar panels. They used batteries only as a small 5-minute energy buffer.

Blink faces the energy variability we study in this paper. Blink tackles it by modulating the motherboards' duty cycle, as it is completely off the electrical grid. In contrast, we study approaches that manage the workload (when it is deferrable or there is a grid outage) and energy sources, including larger batteries and the grid.

HP Labs has recently described a datacenter that is partially powered by a PV solar array [3]. The datacenter is also grid-tied, so that the grid can compensate for the lack of solar energy. For cooling, the datacenter uses outside air combined with standard air conditioners. They focused solely on workload management.

Although their datacenter is conceptually similar to Parasol, they only experimented with a small part of it, 4 servers. In addition, their workload management did not consider grid energy costs, peak grid power costs, or having batteries and the grid at the same time. Without considering these factors, they could not

estimate the total cost savings that green datacenters can provide. Also importantly, their paper does not include any details about their workload management models/algorithms.

Managing workloads in green datacenters. Researchers have proposed to schedule batch jobs to maximize the use of renewable energy [11, 12, 24]. In contrast, Krioukov *et al.* proposed to adjust the quality of the replies provided to users in interactive workloads [25]. For datacenters that run a mix of interactive and batch workloads, Aksanli *et al.* proposed to adapt the amount of batch processing dynamically [2]. Liu *et al.* took the same approach [29]. In addition, they used modeling of the energy cost and the revenue from running batch jobs to derive their workload scheduling.

We consider deferrable (e.g., batch) and non-deferrable (e.g., interactive) workloads in green datacenters. Our management of deferrable workloads is similar to that in [11, 12], as we study the same types of electricity costs. However, for this type of workload, our decisions select the energy source in tandem with the workload schedule. For non-deferrable workloads, our management simply selects the energy source. These previous papers did not consider active energy source management, e.g. the renewable energy is always fully used when enough load is available, and wasted otherwise. Moreover, none of these papers considered net metering, and only [12] considered peak grid power costs.

Managing energy sources in datacenters. Researchers have explored the use of batteries in datacenters [13, 14, 22, 40, 45]. These studies show that batteries and other forms of energy storage can lower both the capital cost of the power delivery infrastructure and the operating cost of a datacenter.

However, none of these papers considered renewable energy or net metering. In addition, almost in their entirety, these studies focused on non-deferrable workloads with available grid power. We consider both deferrable and non-deferrable workloads, as well as grid outages. Interestingly, our results suggest that batteries may be less effective at lowering operating costs in the presence of solar energy and net metering, especially for non-deferrable workloads.

Finally, Li *et al.* have proposed an architecture in which two sets of servers draw power from different sources (e.g., from the grid or from a wind farm) [28]. In their setup, energy source management translates into migrating workloads between the two sets. We argue that their architecture is less flexible and incurs more (performance and energy) overheads than that of Parasol.

3. Tradeoffs in Greening Datacenters

In this section, we discuss the tradeoffs, space requirements, and capital costs involved in greening datacenters, using data from numerous sources. We start by comparing co-location and self-generation to grid-centric approaches. We then discuss how the space requirements and costs of solar and wind have evolved over time, and our expectations for the future.

Grid-centric vs co-location/self-generation. Grid-centric approaches include power purchase agreements and off-site generation contracted by the datacenter operator [34]. In these approaches, the renewable energy is generated at locations with an abundance of the renewable fuel (e.g., windy areas of the midwest in the US), and fed into the electrical grid. Another advantage is that the datacenter operator need not operate or maintain the renewable power plant. However, the power conversions (DC/AC), voltage transformations, and transmission over long distances may incur energy losses of up to 15% [20]. Moreover, in off-site generation, the datacenter operator may incur grid-transmission charges (e.g., the price of 5% of the produced energy) imposed by power utilities [34]. Further, in grid-centric approaches, the datacenter is completely dependent on the availability of the grid or on diesel generators. Grid outages occur frequently in developing countries (e.g., India)

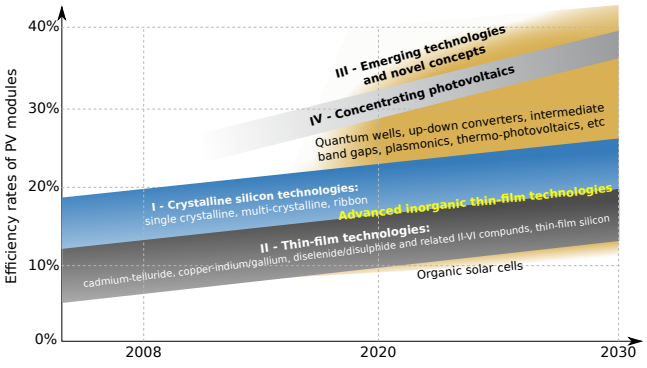

Figure 1. Solar PV efficiency over time. Reproduced from [21].

and in locations of developed countries that are prone to natural disasters (e.g., winter storms in the East of the US).

Co-location and self-generation incur much lower losses (5% or less), as the power undergoes fewer conversions/transformations and is not transmitted over long distances. There are also no additional charges and a lower dependence on the grid or diesel generators (although they do require batteries and/or a grid-tie, for when energy is needed but not enough renewable energy is available). Unfortunately, these approaches do have drawbacks. In co-location, the location of the renewable plant may not be ideal for a datacenter. In self-generation, the location that is ideal for the datacenter may not be so for the renewable plant. Moreover, in self-generation, the datacenter operator is responsible for the operation and maintenance of the renewable plant.

As aforementioned, previous studies [11, 12] have found that self-generation can lower overall electricity costs, as its capital costs can be amortized relatively soon. Comparing self- and off-site generation in terms of electricity costs, a recent study [34] found that the best approach depends on the carbon reduction target.

The above discussion suggests that no approach is perfect, so different operators may make different choices (e.g., Google and Apple). In terms of research, self-generation with solar and/or wind has become a hot topic, as discussed in the previous section.

Solar and wind are two promising sources of green energy for datacenters, as (1) they are more broadly available world-wide than hydroelectric energy; (2) they cause less environmental disruption than hydroelectric energy; and (3) they do not have the waste storage problem of nuclear energy. However, solar and wind have traditionally incurred large space requirements and capital costs.

Space and cost of solar energy. A key factor determining the space requirements of solar PV energy is its efficiency, i.e. the percentage of the sunlight energy that is transformed into electricity. Figure 1 plots the evolution of the efficiency of the different solar PV technologies over time [21]. Note that the efficiency of today's most affordable PV technology (multi-crystalline silicon) hovers around 15% and may increase to 25% by 2030. With other technologies, efficiency is likely to exceed 40%. This future increase in efficiency would represent a 50+% reduction in space requirements.

Another key issue is the "capacity factor", i.e. the percentage of the maximum theoretical solar energy production (24hrs of maximum sunlight every day) that is actually produced. Capacity factors vary depending on latitude and weather. For example, Erfurt (Germany), Princeton (US), Canberra (Australia), and Phoenix (US) have capacity factors of roughly 12.5%, 15.5%, 16.5%, and 24%, respectively [33]. A location with a capacity factor of 24% requires half of the space of a location with a 12% factor.

To make the discussion concrete, consider an example datacenter in which each rack consumes 8.8kW (40 200W 1U servers and a PUE of 1.1), the average power utilization is 33%, and the datacen-

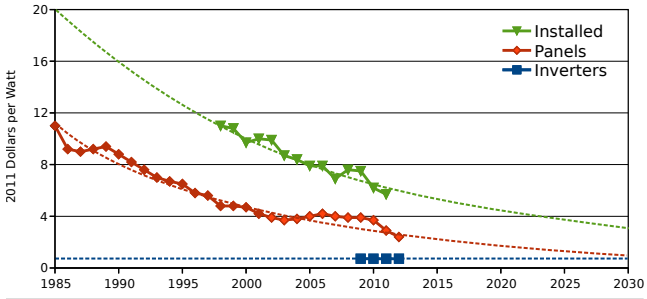

Figure 2. Solar PV capital costs over time without incentives [38, 41]. The data points represent actual average costs, whereas the dashed lines are our approximations of the historical trends. Our cost estimates for the distant future are admittedly speculative.

ter operator wants to produce 50% of the energy it needs from solar in New Jersey (15.5% capacity factor). Moreover, note that the area of a 235W DC solar panel is 17.64 square feet, and it produces a maximum of 202W AC after the de-rating we see in Parasol. Under these conditions, the amount of space required would be roughly 47x larger than the space taken by the racks (each standard rack takes 5.81 square feet, but we multiply this area by 3 to account for clearences in the front and back of the rack). The full calculation is $47 = 8800W/(202W \cdot 15.5\%) \cdot 17.64ft^2/17.43ft^2/(1/33\% \cdot 1/50\%)$. In the 2020-2030 range, we expect these requirements to decrease to roughly 24x. For racks of energy-efficient servers like those of Parasol (80 25W 0.5U servers and a PUE of 1.1), the space requirements are roughly 12x (now) and 6x (future). These requirements seem manageable at least by small/medium datacenters. For example, our predictions suggest that powering 10 racks of our energy-efficient servers (under the assumptions above) will likely require covering only 6 parking spaces with solar panels.

In terms of capital costs, solar energy has become substantially cheaper over time. Figure 2 plots the average final installed capital cost per Watt, and the average capital cost of inverters and panels, over time [38, 41]. The installed cost includes the panels, inverters, and all soft costs, such as permits and labor. Installed costs have decreased by roughly 50% since 1998. Governmental incentives reduce these costs even further. For example, current federal and state incentives in the US can lower installed costs by 40-60%.

The figure also plots dashed curves that approximate the historical data. (Admittedly, we are taking liberties here, as these curves reflect historical trends only, rather than expected technology-, market-, or governmental policy-driven cost changes. Nevertheless, the curves are accurate for the available data.) According to them, the installed cost could decrease by almost 50% by 2030. These trends are in stark contrast to the average cost of grid energy in the US, which has increased by 30+% since 1998 [43].

Section 6 quantifies these costs by computing how long it would take to amortize the installed capital cost of a solar array. Specifically, our experiments suggest amortization periods between 4.8 and 7.1 years, assuming governmental incentives of 60%. In the 2020-2030 time frame, these periods could be less than 2.4 and 3.6 years, assuming that incentives and grid electricity prices remain at the same levels as today. Obviously, these amortization period estimates for the distant future are highly speculative at this point.

Space and cost of wind energy. When a single turbine suffices, wind has lower space requirements than solar. For example, a 1MW wind turbine requires 11000 square feet [32]. In addition, wind has a higher capacity factor than solar in reasonably windy places. For example, the average wind capacity factor for the East of the US is 26%, whereas in the Mountain states it is 33% [42]. Assuming the East capacity factor, 1MW of wind requires roughly 12x less space than 1MW of solar. However, in larger installations, the wind turbines must be placed far apart, causing their overall space requirements to substantially exceed those of solar [30].

In terms of capital costs, wind is 2x-3x cheaper than solar [18].

Despite its small size, Parasol only uses solar energy because winds are not strong enough at our location.

Summary. The above data and trends suggest that self-generation using solar and/or wind energy will become increasingly attractive.

4. Design and Implementation of Parasol

In this section, we describe and justify our design of Parasol as a research platform. We start with the physical infrastructure, then describe its IT hardware and software. At the end of the section, we justify each of our design decisions and mention a few mistakes.

Physical infrastructure. Parasol comprises a small custom container, a set of fixed solar panels, batteries and a grid-tie, a free cooling unit, and a direct-expansion air conditioner (HVAC). The container lies on a steel structure placed on the roof of our building, and houses two 42U racks of energy-efficient IT equipment. Each of our servers takes half-U space, so two racks can host up to 150 of them (some rack space is needed for networking and other gear).

The 16 polycrystalline solar panels are mounted on top of the steel structure and shade the container from the sun most of the time. Each panel produces up to 235W DC. The DC power is transformed into AC using two SMA Sunny Boy 2000HF-US inverters placed inside the container. The panels produce up to 3.2kW of AC power (after derating). Parasol is also equipped with 32kWh of lead-acid batteries controlled by two SMA Sunny Island 5040-US charge controllers. In addition, Parasol is connected to the electrical grid and can be configured to net meter any excess solar energy. In typical on-grid solar setups, the batteries are only discharged during grid power outages. However, we have identified configuration parameters to the inverters and charge controllers that allow us (almost) full control of every source of energy available to Parasol. We can dynamically change the setting of these parameters according to our management goals. As discussed in Section 6, these changes have to be done in stages to prevent instability.

For cooling, Parasol uses the Dantherm Flexibox 450 free cooling unit, TKS 3000 free cooling controller, iA/C 19000 HVAC, and two relays that we introduced (1) to properly coordinate the free cooling unit and the HVAC; and (2) to completely shut down the HVAC when the free cooling is on. Parasol uses free cooling whenever the outside temperature is lower than a programmable threshold (27°C by default). The TKS controller modulates the fan speed according to a temperature sensor inside the container and the threshold. The free cooling unit consumes between 8W and 410W depending on fan speed. When the outside temperature exceeds the threshold, the controller closes the damper, turns the free cooling unit off, and turns the HVAC on. How often the HVAC compressor runs depends on the temperature of its internal sensor and the threshold. The HVAC consumes between 100W and 2.3kW. Under very low temperatures (less than 5°C by default), the HVAC activates its internal 3-kW heater.

To distribute power to the IT equipment, we use 3 Raritan Dominion PX PDUs, which monitor the energy drawn from each outlet and can turn the outlet on/off. Parasol also includes power meters, and temperature, humidity, and air quality sensors.

Figure 3 shows the steel structure, the container, and the solar panels. The steel structure is 8'×22', whereas the container is 7'×12'. The batteries are in the white enclosures on the right side of the photo. The HVAC can be seen on the long side of the container, whereas the exhaust damper can be seen to the left of the container's door. The free cooling unit (not shown) is on the short side of the container in the wall opposite to the damper.

Figure 3. Outside view of Parasol.

Figure 4 shows the layout of the inside of Parasol in scale. The servers face the free cooling unit on the left side of the figure, and expel hot air towards the exhaust damper on the right side. The straight dashed line represents a partition we introduced to create a "cold aisle" in front of the servers. Most of the rest of the container is the "hot aisle". To completely seal the cold aisle, we also placed a partition at the top of the racks (not shown). This top partition leaves the Sunny Boys in the hot aisle. The snake shaped duct brings conditioned air to the cold aisle, when the free cooling unit is off and the damper is closed.

IT hardware. Parasol currently hosts 64 Idotpc Atom-based servers, each of which has a Mini-ITX Motherboard with a dual-core Atom D525MW processor, 4GB of memory, one 250GB hard disk, and one 64GB solid-state drive. Each server consumes between 22W and 30W. To connect the servers, we use 2 low-power 1Gbps Ethernet switches (Cisco SG300-52), each with 52 ports and a power range of 24W to 42W. We will add more of these servers, switches, and PDUs in the near future.

Parasol also includes a quad-core Xeon server with 16GB of memory and 1TB of RAIDed disk space. This server receives and stores all the monitoring data collected from Parasol.

Software. Section 5 describes GreenSwitch. We have also built an energy-management daemon (not used in this paper) for Parasol. The daemon aggressively manages the energy consumption of the IT equipment by sending servers to sleep whenever they are not needed. For example, the daemon transitions inactive servers to the S3 ACPI state after a threshold amount of idle time. Our modified version of the SSH utility wakes up the servers.

Parasol uses Ganglia [35] for collecting and processing all monitoring data. We collect energy consumption or production data from every component (including inverters, cooling units, and PDUs) and quantify the losses in power conversion/distribution. With these data, we compute the Power Usage Efficiency (PUE) at each point in time. We also collect temperature, humidity, air flow, and air quality measurements in different locations. Finally, we monitor the utilization of the servers' CPUs, memories, and disks.

Justification. Our desire to study free cooling (beyond the scope of this paper) is the main reason we place the servers on the roof. Had we placed the servers inside our building, the building's central air conditioning would interfere with our experiments. Based on historical temperature data, we expect Parasol to run on free cooling more than 92% of the time at our location.

We designed the container to be large enough to house a non-trivial number of servers and the auxiliary equipment (inverters and charge controllers), but small enough that it would be cheaper and could be placed just about anywhere. Its internal layout creates a small, sealed cold aisle, and a "wind-tunnel" effect between the free cooling unit, the servers, and the damper. The location of the HVAC is not ideal, requiring the air duct. A better location would have been the opposite wall, near the free cooling unit.

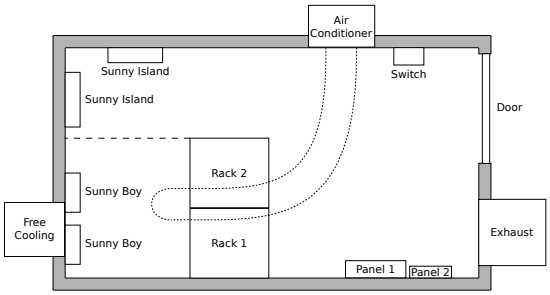

Figure 4. Inside view of Parasol.

The steel frame is larger than the container, so that it can support a larger number of solar panels and the (heavy) batteries. We wanted enough panels to cover at least 30% of the energy consumption of Parasol; through aggressive power management, we expect to achieve this goal easily. The frame is raised to avoid interfering with roof maintenance and the roof's drain system.

Importantly, note that our goal was *not* to design a scaled down version of a warehouse-sized datacenter. Our concept of datacenter is substantially broader, including smaller server installations found in universities and many enterprises, for example. For these installations, it is typically acceptable not to have batteries or backup diesel generators for grid outages. (We included relatively large batteries in Parasol to investigate their use in lowering electricity costs in the face of solar energy and net metering, as we do in this paper.) Moreover, the research we envision on Parasol is independent of most of its physical characteristics. Strictly in terms of its design, Parasol can be seen as a modular datacenter (e.g., [19]) that uses solar energy and free cooling, or as a datacenter for off-grid deployments (possibly in isolated areas).

Our choice of Atom-based servers stems from our desire to study the tradeoffs between wimpy and high-performance servers, as a function of workload (beyond the scope of this paper). For instance, for our Hadoop workloads, our wimpy servers consume 5x less power than comparable Xeon-based servers, while running 4.8x slower. We also considered purchasing ARM-based servers, but those were not yet available at the time.

Finally, our IT equipment runs on AC power, which requires converting the DC power produced by the solar panels and drawn from our batteries. This conversion causes small energy losses (2% on average). We considered a design in which all IT equipment would run on DC power, which would require conversion of the power coming from the grid. Ultimately, we decided that AC power would likely make it easier to find the equipment we wanted.

5. Design and Implementation of GreenSwitch

In this section, we describe GreenSwitch, our model-based framework and system for managing workloads and energy sources in green datacenters. We divide the presentation into two parts: (1) management activities and objectives; and (2) analytical models, and associated optimization goals and solution procedures.

5.1 Management Activities and Objectives

Figure 5 illustrates the GreenSwitch components. The *predictor* predicts the workload and the renewable energy production. The *solver* takes these predictions and the current battery charge level as input, and outputs a workload schedule and an energy source schedule. The solver produces schedules that drive the use of energy by the system. The *configurer* effects the changes prescribed

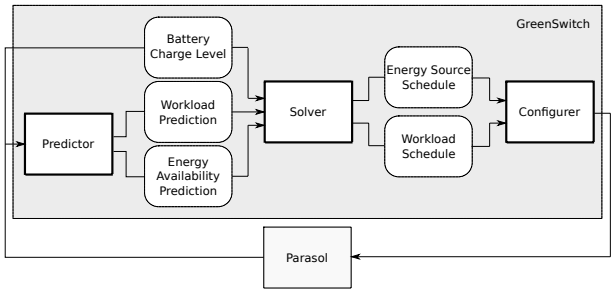

Figure 5. GreenSwitch architecture. Rectangles with round edges are data structures. Rectangles with square borders are processes.

by the solver in the two schedules. The changes may involve transitioning some servers between power states and/or changing the configuration of the energy sources. Monitoring data from the datacenter flows back to inform the predictor in its next iteration.

Note that the configurer is the only part of GreenSwitch that is specific to the system GreenSwitch is supposed to control (e.g., Hadoop); the other components are independent of this system and can be reused without modification. The configurer can be easily adapted to control other systems that provide an energy management interface.

GreenSwitch runs periodically. By default, a full iteration of it occurs every 15 minutes, which enables it to properly control peak grid power consumption. (Power utilities typically compute peak grid power consumption in windows of 15 minutes.) However, GreenSwitch checks the production of solar energy every 3 minutes (by default). During each of these checks, it decides to run a full iteration if there has been an unexpected change in energy production. Specifically, GreenSwitch is conservative and runs a full iteration if it detects a change that is larger than the maximum change we observe in any 15-minute interval on clear days.

Next, we discuss each component of GreenSwitch.

Predictor. The predictor produces predictions of the workload and renewable energy availability for the next "scheduling horizon" (1 day by default, representing the maximum delay a job might experience) at the granularity of shorter "epochs" (1 hour by default, which is the granularity of our solar predictions).

For its **workload prediction**, the predictor can use any method for predicting future loads in terms of the average power they will require in each epoch of the next horizon. Our current implementation assumes that the future will be similar to the past. Specifically, the predictor computes the average power using an exponentially weighted moving average of the average power consumed in the past. It predicts that this average power will be required in every epoch of the horizon. It progressively corrects any inaccuracies in its predictions in its following iterations. Obviously, workloads with repeating patterns could benefit from more sophisticated prediction approaches, e.g. time-series analysis. However, the workloads we have considered so far do not have this property. If necessary, it is easy to replace our current workload prediction implementation with a more sophisticated one.

For its **renewable energy availability prediction**, the predictor can use any model of the availability of solar and/or wind energy. Our implementation uses the method from [12] to predict solar energy. The method combines the model from Sharma *et al.* [36] with the approach for improving accuracy from Goiri *et al.* [11].

Sharma's model recognizes that the solar energy generation is inversely related to the amount of cloud cover, and is expressed 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. We set $B(t)$ to the amount of energy generated on the day with the highest energy generation from the previous month.

Unfortunately, weather forecasts are sometimes wrong, which may lead to inaccurate predictions. To improve accuracy, we apply a second method, which computes $CloudCover$ from the amount of energy generated in the previous epoch. GreenSwitch 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 epoch t, GreenSwitch computes $CloudCover$ for the next epoch using (1) the weather forecast, and (2) the energy produced in epoch $t-1$. At the beginning of epoch $t+1$, it compares the accuracy of the two methods and uses it to predict the remainder of the horizon. At time $t+2$, the process repeats.

Solver. The solver uses the predictions, information on the amount of energy stored in the batteries, and the models detailed in Section 5.2 to produce a workload schedule and an energy source schedule.

The solver can handle two metrics (grid electricity cost and performance degradation), two types of grid electricity cost (energy and peak power), two types of workload (deferrable and non-deferrable), three energy sources (solar, battery, and grid), two grid availability scenarios (grid is always up and grid goes down during the execution of the workload), and three energy storage scenarios (net metering only, batteries only, and batteries plus net metering).

The optimization metrics deserve more explanation. The grid electricity costs reflect the types of charges that utilities may impose. Datacenters often pay a different grid energy price, i.e. a dollar amount per kWh of consumed grid energy, during an off-peak period (e.g., at night) than during an on-peak period (e.g., daytime). In addition, datacenters often pay for their peak grid power consumption, i.e. a dollar amount per kW of grid power at the highest period of grid power usage. To compute the peak charges, utilities typically monitor the average grid power consumption within 15-minute windows during each month. They define the maximum of these averages as the peak grid power for the month.

The performance degradation metric assesses how much of a workload the system has processed over time.

The metric we seek to optimize depends on the workload and the availability of grid electricity. Under deferrable workloads, when the grid is up, we schedule the workload and the use of the energy sources to minimize the grid electricity cost. When the grid goes down, we still schedule both workload and energy sources, but seek to minimize the performance degradation (compared to having the grid up) during the outage. Under non-deferrable workloads, our management objective becomes restricted to scheduling the use of the energy sources to minimize the grid electricity cost.

Workload and energy source schedules. The workload schedule determines how much energy should be consumed in each epoch. For non-deferrable workloads, the workload schedule always matches the workload prediction, so that jobs are not delayed due to GreenSwitch. For deferrable workloads, GreenSwitch has more flexibility to spread the computation around, as long as jobs can still complete within their time bounds.

The energy source schedule determines how much energy to draw from each source in each epoch.

Configurer for Hadoop. In our implementation, GreenSwitch manages a slightly modified version of Hadoop. The modification enables energy management by implementing three server power states: active, decommissioned, and sleep. Active and sleep (the S3 state) are self-explanatory. The decommissioned state is an intermediate state that prevents new jobs from starting on the server. We configure our Hadoop setup to the Covering Subset scheme [27], i.e. we store a full copy of the dataset on the smallest possible

Parameter	Meaning	Default Value
$Load_t$	Average required IT power	N/A
$Workload_t$	Average offered IT load (power)	N/A
T	Length of the scheduling horizon	24 hours
$MaxPower$	Maximum IT power (servers and switches)	64 × 30W + 2 × 42W
$CoverPower$	Typical power consumption of the Covering Subset and switches	8 × 28W + 30W + 24W
$LoadGreen_t$	Amount of green power to be used for the load	N/A
$LoadBrown_t$	Amount of grid power to be used for the load	N/A
$LoadBatt_t$	Amount of battery power to be used for the load	N/A
$BattGreen_t$	Amount of green power to be used for charging batteries	N/A
$NetGreen_t$	Amount of green power to be used in net metering	N/A
$Brown_t$	Amount of grid power to be used for any purpose	N/A
$BattBrown_t$	Amount of grid power to be used for charging batteries	N/A
$CapBatt_0$	Battery charge level in the first epoch	85%
$Duration$	Length of each epoch	15 minutes
β	Energy losses in batteries	10%
$BattMaxCharge$	Maximum charge rate of batteries	8kW
$MaxCapBatt$	Maximum battery capacity	32 kWh
γ	Minimum battery charge level	65%
$BEnergyPrice_t$	Grid energy price	$0.08/kWh (off-peak), $0.13/kWh (on-peak)
$BPeakPrice_t$	Peak grid power price	$13.61/kW/month
α	Percentage of retail price paid in net metering	40%
$PreviousPeak$	Maximum grid power consumed so far	N/A
$MaxPeak$	Maximum peak grid power	N/A
$Load_t^{up}$	Average required IT power when the grid is up	N/A
$Load_t^{down}$	Average required IT power to be used when the grid is down	N/A

Table 1. Model parameters, meanings, and default values. N/A = not applicable.

number of servers; any server out of the Covering Subset can be sent to sleep without affecting data availability.

The configurer has two settings: server energy management on and off. When energy management is off, the configurer does not power-manage any servers. When energy management is on, the configurer abides by the workload schedule by transitioning servers between the three Hadoop server power states. Specifically: (1) it transitions any active server that need not be active but still stores data required by running jobs to the decommissioned state. In a following iteration, if the data stored at the decommissioned server is no longer needed, the configurer transitions it to the sleep state; (2) it transitions any active server that need not be active and does not store relevant data to sleep state; and (3) it transitions sleeping servers to the active state if they are required for computation during an iteration. The configurer keeps the Covering Subset active at all times.

To abide by the energy source schedule, the configurer uses the following actuators: activate/deactivate grid; change maximum current drawn from the grid (which affects the maximum power drawn from the grid); and start/stop charging the batteries. These actuators represent different settings of the parameters of the Sunny Island charge controllers. Interestingly, these actuators must be used carefully to avoid instability. For example, one cannot quickly transition from battery charging with solar energy to net metering, as this causes the inverter to shut down. Thus, this transition needs to be performed in steps, with some idle time in between the steps. Note that *such behaviors are simply overlooked in simulation or with simple prototypes* that do not include real charge controllers.

5.2 Models, Optimization, and Solution Approach

We now describe the solver's models of workload and energy sources. Next, we formulate the optimization problems that the solver implements, and discuss how it instantiates and solves them. Table 1 lists the models' parameters, meanings, and default values.

Modeling workloads. Let us start by defining $Load_t$ to be the average IT power the datacenter will consume in epoch t. (We leave the modeling and management of the cooling power for future work, since we only recently completed Parasol's cooling setup.) We also define $Workload_t$ as the amount of offered computational load (measured in terms of average power per epoch, including the Covering Subset) in epoch t.

We should execute the full workload during the horizon (T), so

$$\sum^{t\in T} Load_t = \sum^{t\in T} Workload_t \quad (1)$$

We can be more specific about this relationship, depending on the type of workload. When the workload is non-deferrable, the offered load must be executed without delay, i.e.

$$\forall t \in T : Load_t = Workload_t \quad (2)$$

When it is deferrable, the offered load can be delayed within T

$$\forall t \in T : \sum_{t'=1}^{t'=t} Load_{t'} \leq \sum_{t'=1}^{t'=t} Workload_{t'} \quad (3)$$

There are also a couple of constraints on $Load_t$. First, the average power during an epoch cannot be higher than the maximum power the load can consume ($MaxPower$), i.e.

$$\forall t \in T : Load_t \leq MaxPower \quad (4)$$

In addition, the power consumption has to be at least that of the servers in the Covering Subset ($CoverPower$), i.e.

$$\forall t \in T : Load_t \geq CoverPower \quad (5)$$

Relating the workload modeling back to the inputs and outputs to the solver, $Load$ represents the workload schedule that the solver produces, whereas $Workload$ is the workload prediction that GreenSwitch feeds to the solver.

Modeling energy sources. In the most general scenario, three sources can be used to power the system: renewable ($LoadGreen$), grid ($LoadBrown$), or battery ($LoadBattery$). Thus,

$$\forall t \in T : Load_t = LoadGreen_t + LoadBrown_t + LoadBatt_t \quad (6)$$

The renewable power ($AvailGreen$) may be used for multiple purposes: running the workload ($LoadGreen$), charging the batteries ($BattGreen$), and/or net metering ($NetGreen$). Thus,

$$\forall t \in T : LoadGreen_t + BattGreen_t + NetGreen_t \leq AvailGreen_t \quad (7)$$

Similarly, the grid ($Brown$) can be used to power the workload ($LoadBrown$) and/or charge the batteries ($BattBrown$).

$$\forall t \in T : LoadBrown_t + BattBrown_t = Brown_t \quad (8)$$

The modeling of the batteries is more involved. First, we must account for the fact that batteries incur losses, i.e. only a percentage (β) of the energy we store in them will be available for later use. We define the current capacity of the batteries ($CapBatt$) as:

$$\forall t \in T : CapBatt_t = CapBatt_{t-1} + Duration \cdot \beta(BattGreen_{t-1} + BattBrown_{t-1}) - Duration \cdot LoadBatt_{t-1} \quad (9)$$

where $Duration$ is the length of each epoch t.

Second, we must formalize the constraints on the batteries. We cannot use more energy than what they have stored, so

$$\forall t \in T : Duration \cdot LoadBatt_t \leq CapBatt_t \quad (10)$$

They have a maximum charge rate ($BattMaxCharge$) in kW:

$$\forall t \in T : BattGreen_t + BattBrown_t < BattMaxCharge \quad (11)$$

They have a maximum capacity ($MaxCapBatt$):

$$\forall t \in T : CapBatt_t \leq MaxCapBatt \quad (12)$$

Third, we must ensure that leveraging the batteries as an energy source does not harm their lifetime. For this, we follow the analysis of Govindan *et al.* [14], which demonstrates that one can avoid reducing the batteries' lifetime by limiting the depth of discharge to a percentage ($CapBatt_0 - \gamma$) of the maximum capacity at all times. (We derive the proper γ in Section 6.) Thus,

$$\forall t \in T : CapBatt_t \geq \gamma MaxCapBatt \quad (13)$$

Fourth, we must account for operational modes that are not supported in typical solar installations. Specifically, we cannot charge the batteries with solar energy and do net metering at the same time.

$$\forall t \in \{T | BattGreen_t > 0\} : NetGreen_t = 0 \quad (14)$$

In addition, we cannot use the batteries and do net metering at the same time.

$$\forall t \in \{T | LoadBatt_t > 0\} : NetGreen_t = 0 \quad (15)$$

We cannot draw from the grid to power the load at the same time as doing net metering.

$$\forall t \in \{T | LoadBrown_t > 0\} : NetGreen_t = 0 \quad (16)$$

We cannot charge and discharge the batteries at the same time.

$$\forall t \in \{T | LoadBatt_t > 0\} : BattGreen_t + BattBrown_t = 0 \quad (17)$$

Finally, we must ensure that all parameters are ≥ 0.

Relating the energy source modeling back to the solver, $AvailGreen$ (renewable energy prediction) and $CapBatt$ (current battery capacity) are inputs, whereas $LoadGreen$, $LoadBrown$, $LoadBatt$, $BattGreen$, $NetGreen$, and $BattBrown$ represent the energy source schedule that the solver produces as an output.

Modeling the peak grid power usage. We define $PreviousPeak$ as the maximum grid power (averaged over 15-minute intervals) consumed from the beginning of the current month until the start of the current horizon. We also define $MaxPeak$ as the highest grid power consumption ($Brown$) since the beginning of the current month, i.e. $Brown$ and $PreviousPeak$ must be no higher than $MaxPeak$.

$$\forall t \in T : Brown_t \leq MaxPeak \quad PreviousPeak \leq MaxPeak \quad (18)$$

Optimization goals. Under the constraints above, we now define our optimization goals: #1 minimize the grid electricity cost, when the grid is up; and #2 minimize the performance degradation (compared to the execution with the grid up), when the grid is down.

To model goal #1, we recognize that the cost of the grid electricity may have both energy and peak power components. We define the energy price per kWh as $BEnergyPrice$, and the peak power price per kW as $BPeakPrice$. When we feed green energy into the grid, we get paid a percentage (α) of the current retail price of the energy by the utility, i.e. $\alpha BEnergyPrice$. For example, the utility may pay the full retail price ($\alpha = 1$) or just the wholesale price ($\alpha < 1$). Thus, we model the first goal as:

$$min(\sum^{t \in T}(BEnergyPrice_t \cdot Duration \cdot Brown_t - \alpha BEnergyPrice_t \cdot Duration \cdot NetGreen_t) + BPeakPrice \cdot (MaxPeak - PreviousPeak)) \quad (19)$$

When the grid is down, goal #1 above does not make sense. In this case, optimization goal #2 becomes active. We model it as:

$$min \sum^{t \in T}(|T| - t)(Load_t^{up} - Load_t^{down}) \quad (20)$$

where $Load_t^{up}$ is the last workload schedule for epoch t before the grid outage, and $Load_t^{down}$ is the workload schedule to be used in epoch t during the outage. Since no new work can reach the datacenter during the outage, we define $Workload_t$ as $Load_t^{up}$ for all t. As the workload and energy predictions tend to be more accurate in the near future, goal #2 minimizes the performance degradation ($Load_t^{up} - Load_t^{down}$) while promoting early load execution ($|T| - t$).

Instantiating the models. Before solving the optimization problems, we need to instantiate the model's parameters. We instantiate them with values from Parasol, GreenSwitch, and our experimental setup. Specifically, we instantiate $MaxPower$, $CoverPower$, $BattMaxCharge$, and $MaxCapBatt$ with Parasol data. In addition, we instantiate $Workload$ with the GreenSwitch workload prediction, and $AvailGreen$ with its solar energy prediction. We also instantiate $Duration$ with the common-case duration of each epoch in GreenSwitch. In our experiments, we study multiple scenarios. When the datacenter is exposed to both grid energy and peak power charges, we instantiate $BEnergyPrice$ and $BPeakPrice$ with real values for those prices. When disregarding peak grid power, we set $BPeakPrice$ to \$0/kW. When considering non-deferrable workloads, we use the constraint defined by Equation 2 above, whereas for deferrable workloads we use Equation 3. To mimic scenarios without batteries, we set $\forall t \in T : LoadBatt_t + BattBrown_t + BattGreen_t = 0$. For scenarios without the grid, we set $\forall t \in T : Brown_t = 0$. For different net metering rates, we adjust α. We set $CapBatt_0$ to an initial battery charge.

Solving the optimization problems. Let us start with the grid electricity cost problem. Although Equation 19 is linear, the solver cannot use simple Linear Programming (LP), because the constraints in Equations 14–17 are non-linear. Fortunately, these constraints can be linearized into Mixed Integer LP (MILP) formulations. We solve the MILP problem efficiently using the Gurobi solver [16].

For goal #2, the solver (1) zeroes all parameters relating to the grid; (2) removes Equation 1, as we may not be able to run the entire workload without the grid; (3) changes γ to 21%. This value is the sum of the battery capacity needed to protect against the largest possible workload misprediction for an epoch (6%, corresponding to the full power of the cluster), and the lowest battery charge level before the charge controllers start to shut down (15%); (4) uses the last workload schedule before the grid outage as the first workload prediction during the outage; and (5) uses MILP to compute the new workload and energy source schedules.

6. Experimental Evaluation

In this section, we describe the remaining details of our methodology and the results of our experiments.

6.1 Methodology

Experimental setup. We run our experiments on Parasol. The 64 servers run our modified version of Hadoop. GreenSwitch runs on one of these servers. We also use a Xeon-based client located in another building to submit the Hadoop jobs. Each of our experiments runs for 23 hours and 15 minutes, starting at 12:00am, so that we have 45 minutes to prepare the next experiment.

Unless we state otherwise, all experiments with solar energy also use net metering. We consider systems with and without batteries. One can think of the system without batteries as a conventional on-grid solar-powered system, i.e. batteries are only used when the electrical grid is out. Another way to think about it is as a system without any batteries whatsoever. When batteries are not used or not present, solar energy deficits are compensated with grid energy, and solar energy surpluses are stored on the grid via net metering.

Our experiments with solar energy are exposed to real weather conditions, making it hard to compare two experiments directly. Fortunately, for non-deferrable workloads, we can first run the GreenSwitch experiment on the real system. Then, using this experimental data, we can manually mimic the no-battery schedule under the same workload and solar energy conditions. We cannot use the same trick for deferrable workloads, because the GreenSwitch schedules interact. For both types of workload, we can compare the GreenSwitch results to a grid-only datacenter setup, i.e. no solar energy, no batteries, and modified Hadoop (which does server energy management, but never defers workloads).

Workloads. We study two widely different Hadoop traces, called "Facebook" and "Nutch". Facebook comes from a larger trace of 600 machines at Facebook [4]. The original trace was collected from May to October 2009, and contains roughly 1 million jobs. We use the Statistical Workload Injector for MapReduce (SWIM) [5] to generate a scaled-down version of the trace for 24 hours on 64 machines. In the resulting trace, each job comprises 2–1190 map tasks and 1–63 reduce tasks. There are roughly 5500 jobs and 68000 tasks. The map phase of each job takes 25–13000 seconds, whereas the reduce phase takes 15–2600 seconds. Jobs have inputs of 64MB–74GB and outputs of up to 4GB. These characteristics lead to a cluster utilization of 27%.

Nutch is the indexing part of the Web search system in CloudSuite [9]. The trace consists of 2000 jobs that index groups of pages previously fetched from our Web domain. Each job runs 42 map tasks and 1 reduce task. Each map phase takes 15–40 seconds, whereas the reduce phase takes 150 seconds. On average, each job touches 85MB of data. Jobs arrive according to a Poisson distribution with mean inter-arrival time of 40 seconds. These characteristics lead to a cluster utilization of 32%.

We run these workloads in the deferrable and non-deferrable modes. The workloads require 8 servers in their Covering Subsets.

Instantiating the models. We use the default values in the third column of Table 1 to instantiate our models. We have already explained all these default values, except for $BEnergyPrice$, $BPeakPrice$, α, $CapBatt_0$, β, and γ.

In terms of electricity prices ($BEnergyPrice$, $BPeakPrice$), we assume on-peak/off-peak pricing, the most common type of variable grid energy pricing. In this scheme, energy costs less during off-peak times (11pm–9am) than during on-peak times (9am–11pm). We use the energy prices charged by the main utility at our location: \$0.13/kWh (on-peak) and \$0.08/kWh (off-peak). This utility charges for peak grid power at \$13.61/kW. By default, we assume that the utility pays the wholesale energy price of 40% (α) of the retail price [43] for net metering. However, we also consider the scenario where the utility pays the retail price ($\alpha = 100\%$).

Finally, we start our experiments with the batteries at 85% of their capacity ($CapBatt_0$), because it takes substantially longer to reach higher charge levels. We computed the battery energy losses (β) from our experiments with Parasol. We calculated the minimum acceptable charge level (γ) from the maximum depth of discharge, using the same approach as [14]. Specifically, we first profiled one year of solar energy production from a local farm to compute the expected number of battery charge/discharge cycles in a system like Parasol. From this data, we find that 90+% of the days would require 2 discharges or less. Assuming a desired lifetime of 4 years, 2 discharges per day translates into 2920 discharges. According to the model in Figure 3b of [14], a maximum depth of discharge of 20% is required to achieve 2920 discharges. So, $\gamma = 85\% - 20\% = 65\%$.

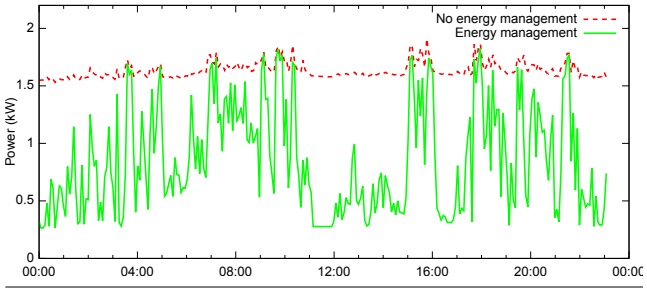

Figure 6. Power with/without energy management for Facebook.

6.2 Experimental Results

This section presents our experimental results. We first compare our modified version of Hadoop to standard Hadoop to establish a baseline. Second, we quantify the accuracy of GreenSwitch's workload and solar energy predictions. Third, we evaluate GreenSwitch's management of the energy sources for a non-deferrable workload. Fourth, we evaluate GreenSwitch's management of workload and energy source for a deferrable workload. Fifth, we study different energy storage scenarios. Sixth, we evaluate GreenSwitch's management of peak grid power draw. Seventh, we study the impact of different amounts of solar energy. Eighth, we consider a different workload. Finally, we show results for a grid outage scenario.

Energy management in Hadoop. We start by comparing the energy consumption of our modified Hadoop to that of the original Hadoop implementation for the Facebook trace in the non-deferrable mode. The modified Hadoop computes the number of active servers to fully pack them with the available map tasks (each server can run 3 maps and 1 reduce concurrently).

Figure 6 shows the power consumed by the trace over time with (modified Hadoop) and without (Hadoop) server energy management. Note that the power consumption without energy management is almost flat, because our servers have a small dynamic power range. The execution becomes much more energy proportional when energy management is in effect. Overall, we find that our modifications to Hadoop enable energy savings of 41%. The savings are higher (roughly 50%) for the Nutch trace in the non-deferrable mode. (We do not consider deferrable workloads in this comparison, because original Hadoop is incapable of leveraging the ability to delay jobs.) Given these positive results, we use our modified Hadoop as the baseline for comparison from now on.

Accuracy of workload and solar energy predictions. To assess the accuracy of our predictions, we compare them to the actual workload and solar energy production for each epoch of the experiments we describe below. These comparisons show average workload prediction errors of 36% 1 hour ahead, and 51% 3 hours ahead, when the workload is non-deferrable. For deferrable workloads, these numbers are 31% and 48%, respectively. In terms of solar energy predictions, these average errors are 13% and 14%, respectively. We do not evaluate the GreenSwitch predictions more

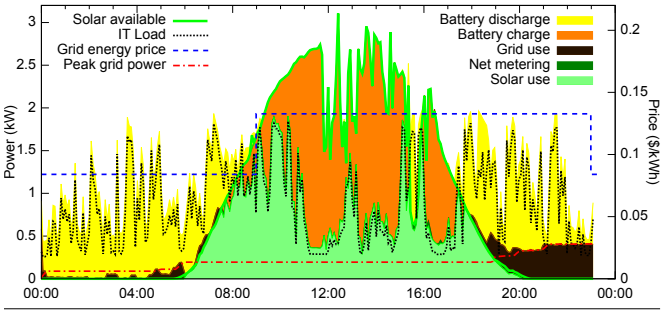

Figure 7. Energy source management for non-deferrable Facebook.

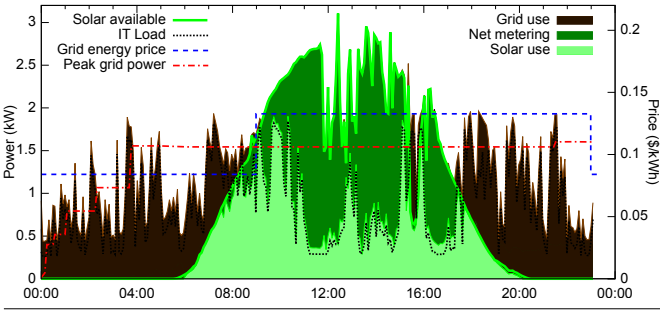

Figure 8. No-battery, non-deferrable Facebook (re-colored).

thoroughly because our methods for producing them are not contributions of our work. Importantly, GreenSwitch is able to quickly detect and adapt to workload and solar energy mispredictions, due to its frequent iterations (3-15 minutes).

Energy source management for a non-deferrable workload. Figure 7 illustrates the GreenSwitch execution of the Facebook trace in the non-deferrable mode on July 17, 2012. The fill colors represent the use of the different energy sources, whereas the lines going across the figure are the solar energy production (full), the IT load (dots), the grid energy price (dashes, Y-axis on the right), and the current peak grid power draw (dashes-dots). Recall that GreenSwitch seeks to minimize the grid electricity cost when the grid is up, and that GreenSwitch schedules the entire workload within a maximum of 1 horizon.

First, we can see that the amount of produced solar energy varied significantly during the day, especially 11am–2pm and 3pm–4pm, due to cloudy conditions. Second, the solar energy was enough to power the workload and charge the batteries. Despite their energy losses, GreenSwitch decided to charge the batteries to attempt to avoid using grid energy while it was still expensive (until 11pm). Third, when there was no solar energy, GreenSwitch drew energy from both the batteries and the grid in many cases. (Recall that GreenSwitch prevents the battery charge level to fall below 65% for lifetime reasons.) Early on, GreenSwitch predicted that it would not have enough battery energy to run the load until solar energy was available. Consequently, it sought to consume grid energy. At night, it again predicted not to have enough battery capacity, and resorted to grid energy. Throughout the day, GreenSwitch was able to limit the increase in the peak grid power by using batteries and/or solar energy. However, it was unable to prevent a slight increase of the peak grid power, as its control of the grid draw (done through a limit on current) is not accurate enough. Note that *this effect would not be seen under simulation or with a simple prototype* that does not include a real grid-tie.

To quantify the electricity cost savings, we compare the use of the energy sources in this figure to those when batteries are not used or not present at all (Figure 8). Because of the difficulty in comparing experiments (Section 6.1), Figure 8 simply re-colors Figure 7 with the colors we would see in the absence of GreenSwitch.

Comparing the executions shows that GreenSwitch reduced the grid energy cost by 73%, the peak grid power cost by 78%, and the overall grid electricity cost by 75%. Despite these high savings, it may be difficult to amortize the capital cost of the batteries ($0.21/Wh in the US [38]) during their lifetime. Specifically, GreenSwitch discharged the battery once per day (to the maximum depth of discharge) in this experiment, extending the batteries' lifetime to 8 years. Under these conditions, the cost of the batteries would have to be roughly 1.8x lower for us to amortize it in 8 years.

A more positive comparison is between a solar-powered system without batteries (same behavior as in Figure 8) and a grid-only datacenter. This comparison shows that the electricity cost savings accrued by the solar-powered system (66%) could amortize its installed capital cost ($2.28/W with incentives) in 7 years.

Workload and energy source management for a deferrable workload. We now consider the GreenSwitch behavior assuming that the jobs in the Facebook trace are deferrable. Figure 9 shows this behavior on July 1, 2012 in the same format as the previous figures. The white filling represents solar energy that was produced but lost due to inefficiency.

The figure shows a significant drop in solar energy production between 4pm and 6pm. We can see also that the energy consumption was low when solar energy was not available. The reason is that deferring load allows GreenSwitch to send more servers to the sleep state; only the servers in the Covering Subset (and the switches) stay active. When there was no solar energy, GreenSwitch drew energy from the batteries, since they stored enough capacity for the load. Another observation is that the available solar energy was enough to power the workload, charge the batteries, and do significant net metering. In the deferrable mode, the workload consumes less energy due to batching effects (better utilization of fewer active servers). The amount of net metering was high for this workload, because it took a relatively small amount of solar energy to fully recharge the batteries. A final observation is that losses (e.g., inverters, PDUs) were significant at lower power consumption, e.g. at night. This is why the fill color goes higher than the load curve in the figure. The same effect can be seen in other figures but it is not as pronounced as when the load is low. Again, *this effect is typically overlooked* (in favor of constant losses) in simpler setups.

As mentioned above, we cannot compare two experiments with solar energy directly. However, we can compare the results of this experiment to those of a grid-only datacenter. This comparison shows that the solar setup with batteries produced a *profit* of 9% in grid electricity cost, deriving from GreenSwitch's ability to intelligently schedule workload and energy source use. Since there was only 1 discharge (down to 70% charge), the batteries' lifetime would be roughly 9.5 years. GreenSwitch would be able to amortize the cost of the solar setup and batteries in 7.6 years.

When we run this workload with solar energy but without batteries (on June 30, 2012), GreenSwitch was able to reduce the grid electricity cost by 96%, compared to a grid-only datacenter. We could amortize the installed capital cost of this setup in 4.8 years.

Since deferrable loads illustrate the ability of GreenSwitch to manage workloads (as well as energy sources), we use this type of workload from now on unless we explicitly state otherwise.

Energy storage approaches. We now turn to investigating the GreenSwitch behavior under different energy storage assumptions. We run experiments with (1) net metering (utility pays wholesale prices) but no batteries; (2) batteries but no net metering; and (3) both batteries and net metering (utility pays retail prices). The behavior of net metering with retail prices but no batteries is that in experiment (1). Recall that the results we presented above (Figure

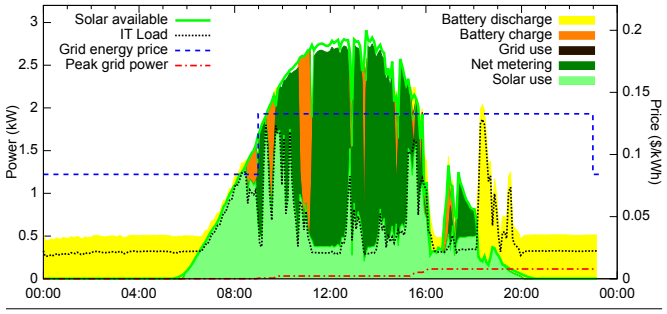

Figure 9. Workload and energy source management for deferrable Facebook.

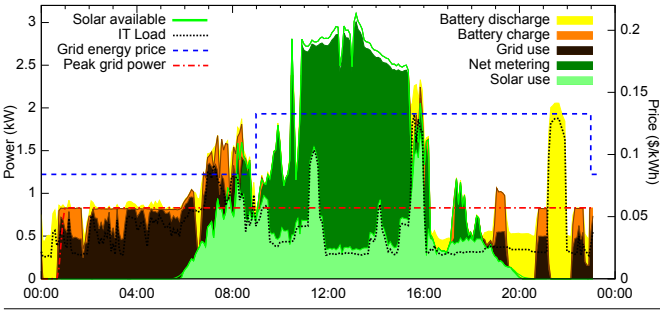

Figure 10. Net metering pays retail energy prices for deferrable Facebook.

7) assume the remaining option: both batteries and net metering (utility pays wholesale prices).

Experiment (1) exhibits the obvious behavior, i.e. GreenSwitch delayed the workload until solar energy become available. Any excess solar energy got net metered. When solar energy was not available, GreenSwitch kept only the servers in the Covering Subset (and switches) in the active state, and used grid energy.

Experiment (2) is more interesting. Since net metering was not available, GreenSwitch consumed energy from the batteries when solar energy was not enough to power the load. However, GreenSwitch did not delay the entire workload until solar energy was available. As it predicted that it would be able to fully recharge the batteries from solar energy later, it scheduled some of the load for the night time.

Finally, in experiment (3), net metering becomes extremely attractive, especially during on-peak hours. For this reason, GreenSwitch net metered all of the excess solar energy. When solar energy was not available, GreenSwitch consumed grid energy when it was cheap, and battery energy when it was expensive. Figure 10 shows these behaviors. Note that GreenSwitch still had to schedule the use of the grid during on-peak hours. The reason is that it predicted not to have enough battery capacity to handle the load during the night.

Unfortunately, we cannot directly compare the electricity costs of these experiments since they were run on different days (June 30, July 9, and July 10, 2012 respectively). However, it is clear that setups that do not require batteries have a major capital cost advantage. For example, under the same weather conditions, the electricity cost savings from setup (1) would be smaller than from using both net metering and batteries. But the capital cost of setup (1) is much lower, if the datacenter can be down during grid outages.

Peak grid power charges. Another important issue is how effective GreenSwitch is at limiting the peak grid power draw. When it disregards these charges, GreenSwitch is more willing to increase the peak grid power, especially when energy is cheap. For the Facebook trace in the non-deferrable mode, the peak grid draw reached 1854W, when GreenSwitch disregarded the peak grid power charges. When it managed the peak grid draw, the peak reached only 408W. This represents a savings in peak power cost of 78%. In the deferrable mode, the peak grid draw reached 711W when GreenSwitch disregarded the peak charges. When it accounted for those charges, the peak reached only 114W; an 84% savings in peak power cost.

These significant savings demonstrate the importance of explicitly managing this aspect of the electricity cost.

Solar energy availability. To see the impact of different weather conditions, consider Figure 11, which corresponds to the Facebook workload in the deferrable mode on a cloudy day (July 13, 2012). Compare this figure with Figure 9, the same workload on a sunny day. On the sunny day, there was no need to use grid energy. On the

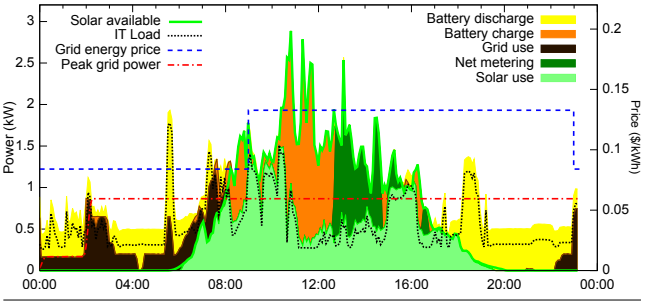

Figure 11. Deferrable Facebook on July 13, 2012.

cloudy day, there was. From its energy predictions, GreenSwitch expected that it would not have enough solar energy to execute the entire workload and charge the batteries. Thus, it decided to use cheap grid energy to compensate. Based on this experiment, the batteries would last 10+ years, whereas it would take 9.2 years to amortize the capital costs of the solar setup and batteries.

We also experimented with the same workload, without batteries, on a cloudy day with a thunderstorm (July 18, 2012). This experiment led to an amortization period of 5.4 years for the solar setup, compared to a grid-only datacenter. An ever worse day (rain the whole day) led to an amortization period of 7 years.

These comparisons show that GreenSwitch is capable of nicely adapting to weather conditions.

Nutch. To demonstrate that our results are similar for widely different workloads, we experimented with Nutch assuming that its jobs are non-deferrable. Figures 12 and 13 show the results with and without energy source management, respectively. Since this workload is in the non-deferrable mode, the latter result is the re-colored version of the original experiment.

These results show that GreenSwitch's energy source management reduced the grid energy cost by 66%, the peak grid power cost by 49%, and the overall grid electricity cost by 59%. As for the Facebook trace, these are significant savings. Also like Facebook, the batteries' lifetime would be 9.5 years. However, their capital cost would have to be 2x lower to be amortized in their lifetimes. Comparing a system without batteries to a grid-only datacenter, we find that the grid electricity cost savings (66%) would amortize the installed capital cost of the solar setup after 7.1 years.

Grid outage. Our final experiment demonstrates the GreenSwitch behavior during a grid outage. Recall that GreenSwitch turns to minimizing performance degradation when the grid goes down. Figure 14 shows the behavior of our system for the Facebook trace in the deferrable mode. We assume that the outage occurs at 12am, following the experiment in Figure 10. Note that the day of the outage (July 19, 2012) was mostly cloudy.

The GreenSwitch workload schedule nicely matched the behavior we would have seen had the grid stayed up. Specifically,

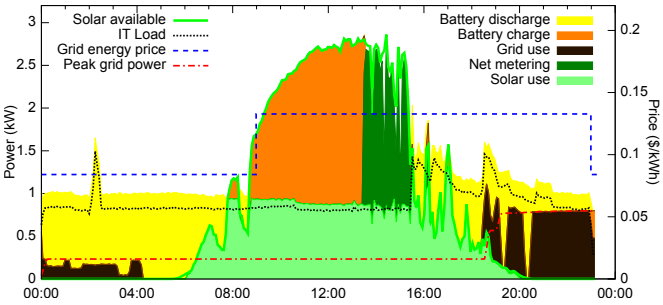

Figure 12. Non-deferrable Nutch workload.

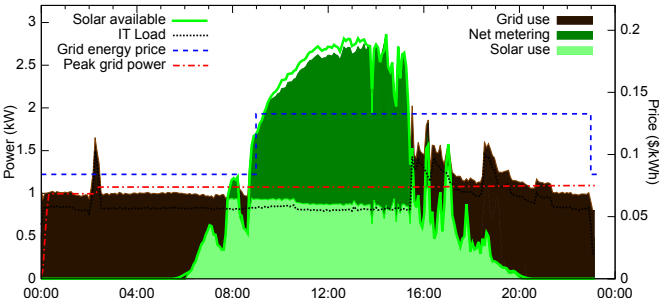

Figure 13. No battery, non-deferrable Nutch (re-colored).

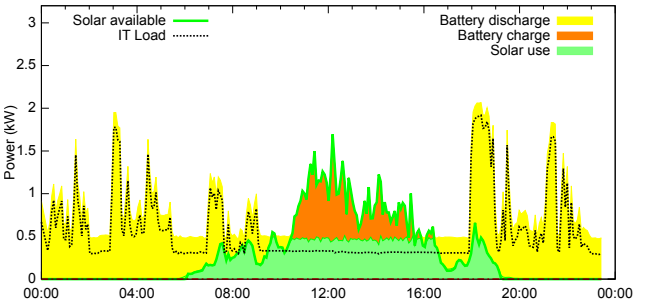

Figure 14. Grid outage on July 19, 2012.

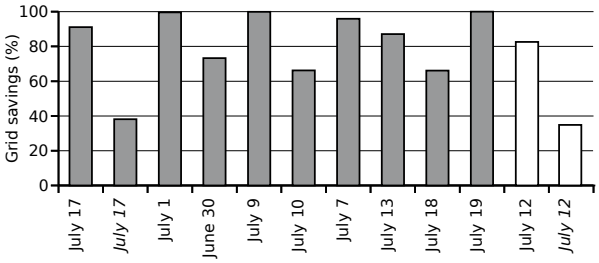

Figure 15. Grid energy reductions compared to grid-only. Repeated date = re-colored result. Gray bars = Facebook; white bars = Nutch.

GreenSwitch intended to execute most of the load early, when the grid energy was cheap, while net metering most of the solar energy for a high retail price. When the grid went down, GreenSwitch kept a similar schedule, executing the entire workload during the day and leaving the battery at 45% capacity.

Reductions in carbon footprint. Figure 15 shows the GreenSwitch reductions in grid energy use compared to a grid-only datacenter for each of the experiments we described above. From left to right, the experiments appear in the order they were discussed above. We identify each experiment by the date when it was run (repeated dates correspond to re-colored results). The two bars furthest to the right are for Nutch, whereas the others are for Facebook. It is clear that GreenSwitch is extremely successful at reducing footprints for both workloads, in some cases completely eliminating any grid energy use (100% reductions). The minimum reduction is 36%.

Summary of results. The results above suggest several interesting observations:

- GreenSwitch's aggressive management of all energy sources (including batteries) during normal operation can produce significant reductions in grid electricity cost and consumption.
- Although GreenSwitch produces the lowest grid electricity costs when batteries are available, their capital cost is a burden for non-deferrable workloads. Specifically, their cost would have to be 1.8x–2x lower to be amortized during their lifetime. Our experiments suggested lifetimes between 8 and 10 years.
- Given the flexibility to delay loads, GreenSwitch can produce integrated workload and energy source schedules that even more significantly lower electricity costs. These schedules enable GreenSwitch to amortize the cost of the batteries, but a solar setup without batteries is still superior.
- The installed capital cost of a solar setup without batteries can be amortized in 4.8–7.1 years in our experiments.
- The GreenSwitch benefits are robust to different renewable energy profiles and workloads. However, every part of GreenSwitch is required, especially its management of peak grid power.
- GreenSwitch is beneficial during grid outages as well, enabling the system to minimize performance degradations.

7. Discussion

Parasol most likely differs from any existing datacenter as: (1) it uses solar energy, batteries, and net metering, whereas most datacenters only draw electricity from the grid and use batteries solely until diesel generators become fully active after a power outage; (2) it uses low-power servers, whereas most datacenters use more powerful servers; and (3) its number of servers (150 max) is comparable to those of datacenters found in universities and enterprises, but tiny compared those of the warehouse-scale datacenters that host popular Internet services.

Despite these differences, our experience and results should be useful in practice for several reasons. First, Parasol exemplifies the type of green datacenter one may choose to deploy at universities, enterprises, or remote locations. Second, GreenSwitch tackles the set of issues and tradeoffs one may face in managing energy sources and workloads in green datacenters of any size. Third, low-power servers are energy-efficient for many workloads (including those in this paper). Moreover, the fact that they incur higher execution times is often not a major concern, especially when the workload is deferrable. Fourth, the electricity cost savings, grid energy savings, amortization periods, and battery lifetimes we report are relative to the size of Parasol. Thus, scaling Parasol (and its workload) proportionally to larger setups (e.g., by doubling the number of servers, the number of solar panels, and the battery capacity) would likely produce similar GreenSwitch results. Obviously, warehouse-scale datacenters are more complex than just a large multiple of Parasol, so the results could be quite different for those systems.

8. Conclusions

In this paper, we presented (1) an analysis of the main tradeoffs involved in powering datacenters with solar and/or wind energy; (2) the design and justification for a research platform, called Parasol, that powers a micro-datacenter with solar energy; (3) a system,

called GreenSwitch, for managing workload execution and energy source use in a datacenter powered by solar energy; and (4) a detailed evaluation of GreenSwitch. We discussed many issues that we faced solely because of our use of a real green datacenter prototype. We conclude that the use of renewable energy in datacenters may become increasingly appealing over time, which will likely encourage more initiatives such as Parasol and GreenSwitch.

Based on our experience with our real systems and to extrapolate from the results we presented in this paper, we are currently implementing a simulator that will accurately mimic Parasol and GreenSwitch for longer periods, e.g. a year. The simulator will also enable comparisons against an "offline optimal" version of GreenSwitch that has complete future knowledge of renewable energy production and workload. In the future, we plan to enable other researchers to use Parasol remotely by dedicating servers to their efforts and virtualizing our energy sources.

Acknowledgements

We would like to thank David Meisner and Anand Sivasubramaniam for comments that helped us improve this paper. We are also indebted to Joan Stanton, Heidi Szymanski, Jon Tenenbaum, Chuck Depasquale, SMA America, and Michael J. Pazzani for their extensive help in building/funding Parasol.

References

[1] AISO.net. Web Hosting as Nature Intended, 2012. http://www.aiso.net/.

[2] B. Aksanli et al. Utilizing Green Energy Prediction to Schedule Mixed Batch and Service Jobs in Data Centers. In *HotPower*, 2011.

[3] M. Arlitt et al. Towards the Design and Operation of Net-Zero Energy Data Centers. In *ITherm*, 2012.

[4] Y. Chen et al. The Case for Evaluating MapReduce Performance Using Workload Suites. In *MASCOTS*, 2011.

[5] Y. Chen et al. Statistical Workload Injector for MapReduce, 2012.

[6] Data Center Knowledge. Apple Plans 20MW of Solar Power for iDataCenter, 2012. http://www.datacenterknowledge.com/archives/-2012/02/20/apple-plans-20mw-of-solar-power-for-idatacenter/.

[7] Data Center Knowledge. Data Centers Scale Up Their Solar Power, 2012. http://www.datacenterknowledge.com/archives/2012/05/14/-data-centers-scale-up-their-solarpower/.

[8] EcobusinessLinks. Green Hosting - Sustainable Solar & Wind Energy Web Hosting, 2012. http://www.ecobusinesslinks.com/-green_webhosts/.

[9] EPFL. CloudSuite, 2012. http://parsa.epfl.ch/cloudsuite/cloudsuite.html.

[10] D. Gmach et al. Capacity Planning and Power Management to Exploit Sustainable Energy. In *CNSM*, 2010.

[11] I. Goiri et al. GreenSlot: Scheduling Energy Consumption in Green Datacenters. In *Supercomputing*, 2011.

[12] I. Goiri et al. GreenHadoop: Leveraging Green Energy in Data-Processing Frameworks. In *EuroSys*, 2012.

[13] S. Govindan et al. Leveraging Stored Energy for Handling Power Emergencies in Aggressively Provisioned Datacenters. In *ASPLOS*, 2012.

[14] S. Govindan, A. Sivasubramanian, and B. Urgaonkar. Benefits and Limitations of Tapping into Stored Energy For Datacenters. In *ISCA*, 2011.

[15] Green House Data. An Economically Responsible Data Center, 2012. http://www.greenhousedata.com/.

[16] Gurobi Optimization, Inc. Gurobi Optimizer 5.0, 2012. http://www.gurobi.com.

[17] J. Hamilton. I Love Solar Power But..., 2012. http://perspectives.-mvdirona.com/2012/03/17/ILoveSolarPowerBut.aspx.

[18] P. Hearps and D. McConnell. Renewable Energy Technology Cost Review. Technical report, Melbourne Energy Institute, 2011.

[19] IBM. Portable Modular Data Center, 2012. http://www-935.ibm.com/services/us/igs/flexible-design.html.

[20] International Electrotechnical Commission. Efficient Electrical Transmission and Distribution. Technical report, 2007.

[21] International Energy Agency. Technology Roadmap – Solar Photovoltaic Energy. Technical report, 2010.

[22] V. Kontorinis et al. Managing Distributed UPS Energy for Effective Power Capping in Data Centers. In *ISCA*, 2012.

[23] J. Koomey. Growth in Data Center Electricity Use 2005 to 2010, 2011. Analytic Press.

[24] A. Krioukov et al. Integrating Renewable Energy Using Data Analytics Systems: Challenges and Opportunities. *Bulletin of the IEEE Computer Society Technical Committee*, 2011.

[25] A. Krioukov et al. Design and Evaluation of an Energy Agile Computing Cluster. Technical Report EECS-2012-13, UC Berkeley, 2012.

[26] K. Le et al. Cost- And Energy-Aware Load Distribution Across Data Centers. In *HotPower*, 2009.

[27] J. Leverich and C. Kozyrakis. On the Energy (In)efficiency of Hadoop Clusters. In *HotPower*, 2009.

[28] C. Li, A. Qouneh, and T. Li. iSwitch: Coordinating and Optimizing Renewable Energy Powered Server Clusters. In *ISCA*, 2012.

[29] Z. Liu et al. Renewable and Cooling Aware Workload Management for Sustainable Data Centers. In *SIGMETRICS*, 2012.

[30] M. Love. Land Area and Storage Requirements for Wind and Solar Generation to Meet the US Hourly Electrical Demand. Master's thesis, Department of Mechanical Engineering, University of Victoria, 2003.

[31] J. Mankoff, R. Kravets, and E. Blevis. Some Computer Science Issues in Creating a Sustainable World. *IEEE Computer*, 41(8), 2008.

[32] National Renewable Energy Laboratory. Wind Farm Area Calculator, 2012. http://www.nrel.gov/analysis/power_databook/calc_wind.php.

[33] PVOutput.org. PV Outputs, 2012. http://www.pvoutput.org.

[34] C. Ren et al. Carbon-Aware Energy Capacity Planning for Datacenters. In *MASCOTS*, 2012.

[35] F. D. Sacerdoti et al. Wide Area Cluster Monitoring with Ganglia. In *Cluster*, 2003.

[36] N. Sharma et al. Cloudy Computing: Leveraging Weather Forecasts in Energy Harvesting Sensor Systems. In *SECON*, 2010.

[37] N. Sharma et al. Blink: Managing Server Clusters on Intermittent Power. In *ASPLOS*, 2011.

[38] SolarBuzz. Solar market research and analysis, 2011. http://www.-solarbuzz.com/facts-and-figures.

[39] C. Stewart and K. Shen. Some Joules Are More Precious Than Others: Managing Renewable Energy in the Datacenter. In *HotPower*, 2009.

[40] R. Urgaonkar et al. Optimal Power Cost Management Using Stored Energy in Data Centers. In *SIGMETRICS*, 2011.

[41] US Department of Energy. 2010 Solar Technologies Market Report. Technical report, 2011.

[42] US Department of Energy. 2010 Wind Technologies Market Report. Technical report, 2011.

[43] US Energy Information Administration. Electricity prices, 2012. http://www.eia.gov/.

[44] US Environmental Protection Agency. Report to Congress on Server and Data Center Energy Efficiency, 2007.

[45] D. Wang et al. Energy Storage in Datacenters: What, Where, and How much? In *SIGMETRICS*, 2012.