Some Joules Are More Precious Than Others: Managing Renewable Energy in the Datacenter*

Christopher Stewart
The Ohio State University

Kai Shen
University of Rochester

1. INTRODUCTION

Large cloud-computing datacenters now host a wide range of business applications, ranging from e-commerce websites to search engines to data mining. Increasingly, these datacenters use renewable energy from wind turbines or solar panels to reduce their dependence on costly and less clean energy from the grid [1, 4, 17]. This paper argues that datacenter management should be revised to maximize the use of such off-grid renewables.[1] Our argument is similar in spirit to the case for energy-efficient management [7, 11, 18]. But efficiency alone may not take full advantage of renewables. For instance, energy-efficient management may batch writes so disks can sleep for long durations. But sometimes renewables are intermittently unavailable, long sleeps may miss opportunities to use them.

Operating systems and cluster management software already manage energy, CPU cycles, and other low-level resources. To be sure, many past techniques will probably carry over to renewables. However, unlike other managed resources, renewables are available only when the wind blows or the sun shines. Such intermittency makes it hard for datacenters to use renewables effectively. For example, windy nights that follow calm afternoons would produce renewables only during traditionally light workload times for datacenters [2, 15]. In the first part of this paper, we model a datacenter that gets power from intermittent wind renewables. Our results highlight challenges in managing an uncontrollable power resource that is intermittently unavailable.

As systems researchers, we believe that coordinated resource management could help datacenters use clean renewable energy, but only if managing uncontrollable and intermittent resources becomes a research priority. Table 1 shows that opportunities to manage renewables more effectively are widespread, ranging from the OS to cluster-wide tools. These opportunities share a common approach to deal with intermittency: they reschedule or migrate work based on the availability of renewables. They also share a common challenge: their policies for scheduling and migration must adapt to uncontrollable changes in the availability and abundance of renewables. For instance, one opportunity (#3) is to send user requests to datacenters that have excess renewables. If the production of renewables at the target datacenter decreases (e.g., lower wind speeds), management must adapt by sending fewer requests. But precisely how many and which user requests should be migrated? The second part of this paper describes our ongoing work on request-level power and energy profiling. Our goal is to provide precise accounting of per-request energy consumption as a guide for budgeting renewables.

This paper outlines a research agenda for managing renewables in the datacenter. Our agenda compliments ongoing efforts to integrate renewables into the grid. However, intermittency presents an even greater challenge for grid workloads, because (for the most part) they can not be rescheduled or migrated. Even today's most integrated grids can satisfy only 19% of their workload from renewable sources [3]. We believe that the unique characteristics of datacenter workloads have the opportunity to further reduce their dependence on non-renewable energy.

2. INTERMITTENCY

In this section, we present a datacenter model in which renewable energy sources, e.g., wind turbines and solar panels, can power a datacenter. When renewables are intermittently unavailable, datacenters must get power

*This work was supported in part by NSF CAREER Award CCF-0448413, grants CNS-0615045, CCF-0621472, and by an IBM Faculty Award.

[1] A renewable is an electrical joule converted from solar/wind energy.

Aspect of Data Center Management	Opportunities to Use Renewables More Effectively	Adjustments During Intermittent Outages
Capacity planning	Power certain machines only when renewables are available.	Turn off some machines.
	Plan the geographic location of datacenter sites according to intermittency patterns.	Connect some sites to the grid.
Load balancing	Route requests to datacenters with unused renewables.	Route fewer requests.
	Move entire services to datacenters that expect long periods of renewable power.	Re-migrate services to other datacenters.
Job scheduling	Aggressively prefetch from HDD before expected outages.	Prefetch less data.
System maintenance	Delay SSD erasures until renewables are available.	Further delay erasures.

Table 1: Opportunities to improve datacenter management of renewables grouped by management function. Time-shifting opportunities delay work until renewables are available. Renewable-aware migration moves work to areas where renewables are available.

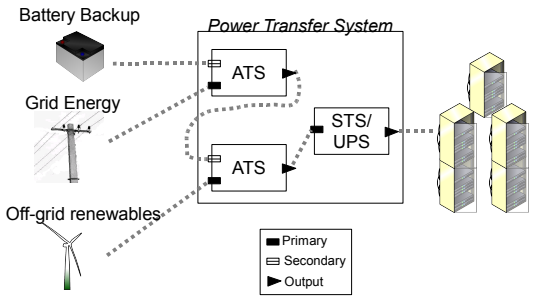

Figure 1: Power transfer system for a datacenter that uses energy from a wind turbine, the grid, and a backup generator.

from other energy sources. A *power transfer system* safely manages energy sources by 1) isolating the electricity from different sources, and 2) ensuring that the datacenter gets enough power. Figure 1 depicts the key elements in a power transfer system. Automatic transfer switches (ATS) are connected to primary and secondary power sources. If power supply from the primary source falls below a preset threshold, the *power-transfer threshold*, the ATS disconnects from the primary source (for isolation) and connects to a secondary energy source. When the primary source can meet the power-transfer threshold again, the ATS disconnects from the secondary source and reconnects to the primary source. An uninterrupted power supply (UPS) ensures continuous power delivery to the actual computing equipment.

The power-transfer threshold affects the use of renewables and system reliability. On one hand, a threshold that exceeds the datacenter's actual power draw may disable capable renewable energy sources. On the other hand, thresholds that are too low can cause brownouts. We believe the power-transfer threshold is an important research problem for the architecture and dependability communities. An ideal architecture would avoid brownouts during periods of high power draw, while using all available renewables as if the threshold were zero.

Some energy sources, like the grid, can provide power any time and on demand, but renewable sources, like wind turbines, can provide power only intermittently. In this sense, renewable joules are truly precious system resources [18], because at times they can be scarce or entirely unavailable. Unfortunately, datacenters operating under light, low-energy workloads may not use every available renewable. Without massive and costly energy storage, these precious joules are lost.

To quantify the effects of intermittency, power-transfer threshold, and workload patterns, we modeled the consumption of renewables for the datacenter in Figure 1. We assumed that the turbine's peak production and the datacenter's peak power consumption were 1 MW. We used 10-minute snapshots of monitored wind turbines to characterize renewable power production. The snapshots came from the National Renewable Energy Lab [12], and were taken from 2004 to 2006. We studied snapshots from the following turbine sites:

- **MT**: Site #26943 produced at peak power for 25% of the sampled snapshots. It produced some power for 42% of the sampled snapshots.
- **CA**: Site #11558 produced at peak power for 16% of the sampled snapshots. It produced renewable power for 27% of the sampled snapshots.

Figure 2 shows that the durations of intermittent outages followed a heavy-tail distribution. Outages that lasted longer than 20 minutes would most likely last longer than 500 minutes. Such long-lasting outages exceed the capacity of affordable, datacenter-scale energy storage. Datacenters with continuous power draw require backup energy sources that can fill such long outages. Figure 2 also shows that the durations of continuous power production were heavy tail too. Long-lasting continuous power production presents a challenge for datacenter management to use every precious renewable,

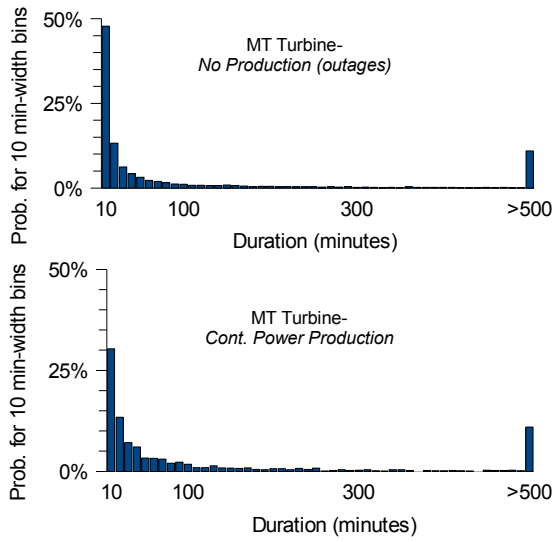

Figure 2: Histograms of outage durations in the MT turbine. An outage occured when wind speeds were below the cut-in or above the cut-out speeds.

even as user-supplied workload fluctuates.

Table 2 shows how many renewables our datacenter would use from each turbine. The columns compare a 1-MW power-transfer threshold, which conservatively guards against brownouts, to a zero power-transfer threshold. The conservative threshold caused a 61–65% drop in the consumption of renewables. A lower threshold allows datacenters to avoid losing these renewables if they can keep their actual power draw below power production levels. Section 3 presents request-level energy/power profiles to manage such changing budgets for renewables.

The rows in Table 2 compare different average datacenter workloads when renewables are available (in KW). Under a conservative threshold, workload increases linearly increased the consumption of renewables. This is expected since the entire workload fits within the threshold. Under a zero power-transfer threshold, workload increases had less than linear effect. This is because power production is not always sufficient to handle the entire workload.

We also assessed the economic feasibility of our hypothetical datacenter. The primary costs for wind turbines and solar panels are capital expenses, offering an opportunity for economies of scale in the production renewables. A 1-MW turbine connected directly to a datacenter would cost about $1.6M USD [8].[2] Maintenance costs are around 2% annually. To outperform the average price for commercial electricity from the US grid ($0.10 KW-hour), the datacenter would need to use 24M

[2]Based on an exchange rate of 1 Euro for $1.34.

		Power-Transfer Threshold	
Loc.	Workload	1 MW	Zero
MT	100 KW	0.24M KW-Hrs	0.68
	400 KW	0.98	**1.92**
	700 KW	**1.72**	**2.89**
CA	100 KW	0.15	0.41
	400 KW	0.63	**1.29**
	700 KW	1.11	**1.93**

Table 2: The consumption of renewables across turbines, workloads, and power-transfer thresholds. Results are in millions of kilowatt-hours. In the bolded scenarios, renewables cost less that $0.10 KW-H.

KW-hours over the turbine's 20-year lifetime. Both the MT and CA turbines could meet this goal under heavy workloads and low power-transfer thresholds. The MT turbine could achieve energy costs as low $0.04 KW-hour.

3. REQUEST-LEVEL POWER/ENERGY PROFILING

Section 2 showed that datacenters risk brownouts when they aggressively utilize intermittent renewable energy sources. To avoid brownouts, it is essential to understand the system power draw levels under different workload conditions. Although workload profiling can be done at different levels, here we focus on the request-level power/energy management. A request is a basic unit of work in most datacenters; it is the aggregate of server system executions in response to an external user demand (*e.g.*, browsing an online catalog or querying database tables for an answer). Requests are a convenient abstraction for dealing with intermittency because they can be distributed on arrival to datacenters where renewables happen to be available.

The measurement of full system energy usage allows calculation of the per-request average (*e.g.*, about 1K Joules on average for a Google search [9]). However, it is challenging to profile fine-grained power draw or energy consumption for individual requests. Part of the challenge lies in the difficulty of identifying request activities in complex server systems. Another part of the challenge lies in modern hardware (e.g., processors, memory, and I/O) that have fluctuating and workload-dependent power demands. Today, a mechanism for fine-grained power measurement is not available from off-the-shelf systems. In the remainder of this section, we present preliminary work on request-level power/energy profiling by combining two existing techniques.

Request-Level Event Profiling.

In a multi-component server system, a request may flow through multiple system components and online

request-level profiling must track these context propagations. Past research [5, 14] has shown that request context propagations can be tracked by analyzing or tagging operating system events. Specifically, techniques developed in our earlier work [14] allow us to collect per-request event statistics, including request CPU usage and hardware event counts available on modern processors. Coupled with an event-driven power model (described next), we can estimate per-request power and energy statistics.

Event-Driven Power Modeling.

Bellosa suggested that fine-grained processor and memory power draw may be estimated using a linear model on hardware event counts [6]. The rationale is that these event counts indicate the intensity of power-relevant activities on system components. Following this rationale, we perform a case study on a set of typical datacenter workloads.

Our case study uses a machine with two dual-core (four cores total) Intel Xeon 5160 3.0 GHz Woodcrest processors. Each core is equipped with two general-purpose event counting registers and a few fixed counters [10]. We configured the performance counters to assemble three predictor metrics for our power model: L2 cache requests per CPU cycle (C_{cache}), memory transactions per CPU cycle (C_{mem}), and the ratio of non-halt CPU cycles (C_{nonhalt}). These metrics indicate the intensity of activities on the L2 cache, memory, and CPU respectively. Formally, the power is modeled as:

$$P_{\text{idle}} + P_{\text{cache}} \cdot \frac{C_{\text{cache}}}{C_{\text{cache}}^{\text{ceil}}} + P_{\text{mem}} \cdot \frac{C_{\text{mem}}}{C_{\text{mem}}^{\text{ceil}}} + P_{\text{nonhalt}} \cdot \frac{C_{\text{nonhalt}}}{C_{\text{nonhalt}}^{\text{ceil}}}, \quad (1)$$

where P's are coefficient parameters for the linear model. $C_{\text{cache}}^{\text{ceil}}$, $C_{\text{mem}}^{\text{ceil}}$, and $C_{\text{nonhalt}}^{\text{ceil}}$ are constants that approximate ceiling values for the predictor metrics. We use microbenchmarks to determine that $C_{\text{cache}}^{\text{ceil}}$=0.1569, $C_{\text{mem}}^{\text{ceil}}$=0.0178, and $C_{\text{nonhalt}}^{\text{ceil}}$=1.0.

We use several workloads to calibrate the model parameters. They include six microbenchmarks: 1) idle; 2) CPU spinning with no access to cache or memory; 3/4) Apache web server with either short requests (no more than 1 KB files) or long requests (files of 100 KB–1 MB); 5/6) OpenSSL RSA encryption/decryption using either a small key or a large key. We also use four full server workloads: 7) TPC-C running on the MySQL database; 8) TPC-H running on the MySQL database; 9) RUBiS [13]; 10) WeBWorK [16]. Since our modeling focuses on the processor and memory power, we avoid disk I/O during the calibration by configuring the datasets to fit in the memory.

Using the least-square fit regression, the model calibration results for Equation 1 are:

- P_{cache} = 8.3 Watts;
- P_{mem} = 20.1 Watts;
- P_{nonhalt} = 27.9 Watts;
- the idle power is 223.0 Watts for the whole system, or 55.8 Watts per-core.

As an indication of high regression accuracy, the result's R^2-statistic is 0.995. In other words, the relative aggregate square error is $1 - R^2 = 0.005$.

Results.

With request-level event accounting and event-driven power modeling, we can construct the power/energy profiles for individual requests. Figure 3 shows the distribution of request power draw for TPC-H, RUBiS, and WeBWorK. The power for a request is calculated as the average power during the request's execution. Figure 4 shows the request energy usage distributions for the three workloads. Results show a small variation on the power draw of different requests. At the same time, the request energy consumption exhibits much larger variations, which are mainly attributed to the varying execution time of different requests.

We demonstrate a promising approach for request-level power/energy profiling. However, the results of our case study is still preliminary while additional validations and experiments in realistic datacenter environments are needed in the future.

4. EXAMPLE SCENARIO

Consider *MakeItRain.com*, a fictional cloud-computing provider that runs large datacenters for Internet services. MakeItRain.com replicates each service across several datacenters for load balancing and fault tolerance. User requests for the hosted services are routed to datacenters in a weighted round robin fashion. To support elastic workload growth, MakeItRain.com must over-provision server resources such that datacenters are not completely used during normal operation.

MakeItRain.com has also invested in energy-related optimizations for its datacenters. First, each datacenter uses PowerNap and a redundant array of power supplies to make the datacenter's actual power draw proportional to its request processing workload [11]. Second, some datacenters have been equipped with renewable sources of energy (like in Figure 1).

This paper makes two key observations related to the management of MakeItRain's datacenters. First, workload distribution (i.e., request routing policies) should be aware of the availability of renewable energy at different datacenters. Datacenters with excess renewables should process a greater share of the user requests, increasing their power draw. The second key observation

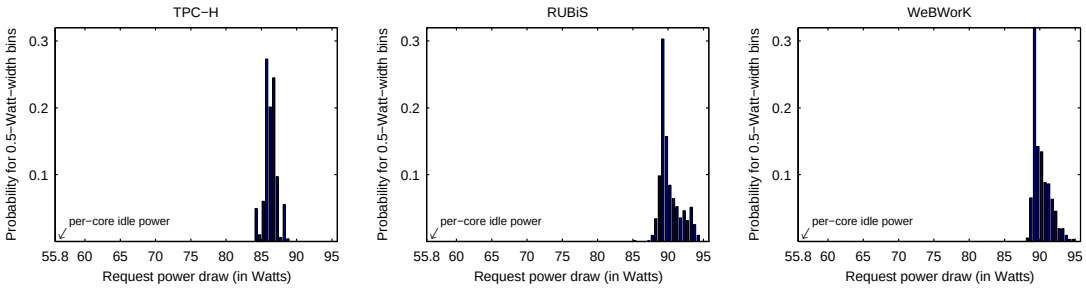

Figure 3: Request power draw distributions in histograms. Results are synthesized from 1000 TPC-H requests, 4000 RUBiS requests, and 1000 WeBWorK requests. For TPC-H, requests are issued for Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q11, Q12, Q13, Q14, Q15, Q17, Q19, Q20, and Q22 in a round robin fashion.

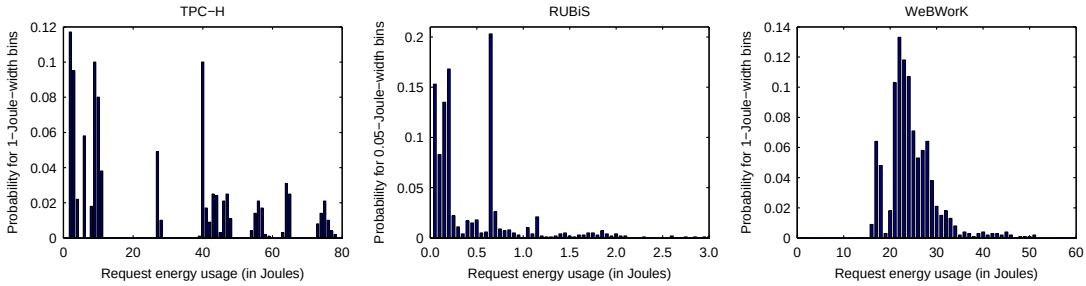

Figure 4: Request energy usage distributions in histograms.

is that it is possible to build request-level power/energy profiles that could *potentially* guide fine-grained request routing. For instance, the profiles could be used to distribute energy-hungry requests, identified by type, to datacenters with excess renewables.

This is just one example of renewable-aware management. Our ongoing work is comprehensively revisiting many aspects of datacenter management to deal with intermittent renewable resources. We will continue to explore renewable-aware request distribution, but we are also investigating other opportunities listed in Table 1. We are also exploring architectural trade offs for low-threshold power transfer systems.

Acknowledgment

We thank Xin Li (University of Rochester) for assisting the power model calibration experiments in Section 3.

5. REFERENCES

[1] Green House Data: Greening the data center. http://www.greenhousedata.com/.

[2] Realistic nonstationary workloads. http://www.cs.rochester.edu/u/stewart/models.html.

[3] Wind power. http://en.wikipedia.org/wiki/Wind_power.

[4] Google solar panel project. http://www.google.com/corporate/solarpanels/home, June 2007.

[5] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier. Using Magpie for request extraction and workload modeling. In *USENIX Symp. on Operating Systems Design and Implementation*, Dec. 2004.

[6] F. Bellosa. The benefits of event-driven energy accounting in power-sensitive systems. In *9th ACM SIGOPS European Workshop*, Sept. 2000.

[7] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle. Managing energy and server resources in hosting centers. In *ACM Symp. on Operating Systems Principles*, Oct. 2001.

[8] European Wind Energy Association. The economics of wind energy. http://www.ewea.org/.

[9] U. Hölzle. Powering a Google search. http://googleblog.blogspot.com/2009/01/powering-google-search.html, Jan. 2009.

[10] Intel 64 and IA-32 architectures software developer's manual volume 3B: System programming guide, part 2, table B-7. http://download.intel.com/design/processor/manuals/253669.pdf.

[11] D. Meisner, B. Gold, and T. Wenisch. Powernap: Eliminating server idle power. In *Int'l Conf. on Architectural Support for Programming Languages and Operating Systems*, Mar. 2009.

[12] National Renewable Energy Laboratory. NREL: Western wind resources dataset. http://wind.nrel.gov/Web_nrel/, 2009.

[13] RUBiS: Rice University Bidding System. http://rubis.objectweb.org.

[14] K. Shen, M. Zhong, S. Dwarkadas, C. Li, C. Stewart, and X. Zhang. Hardware counter driven on-the-fly request signatures. In *Int'l Conf. on Architectural Support for Programming Languages and Operating Systems*, Mar. 2008.

[15] C. Stewart, T. Kelly, and A. Zhang. Exploiting nonstationarity for performance prediction. In *EuroSys Conf.*, Mar. 2007.

[16] C. Stewart, M. Leventi, and K. Shen. Empirical examination of a collaborative web application. In *IEEE Int'l Symp. on Workload Characterization*, Seattle, WA, Sept. 2008. Benchmark available at http://www.cs.rochester.edu/u/stewart/collaborative.html.

[17] P. Thibodeau. Wind power data center project planned in urban area. *ComputerWorld*, Apr. 2008.

[18] A. Vahdat, A. Lebeck, and C. Ellis. Every joule is precious: the case for revisiting operating system design for energy efficiency. In *ACM SIGOPS European Workshop*, Sept. 2000.