Towards Continuous Policy-driven Demand Response in Data Centers

David Irwin, Navin Sharma, and Prashant Shenoy
University of Massachusetts, Amherst
{irwin,nksharma,shenoy}@cs.umass.edu

ABSTRACT

Demand response (DR) is a technique for balancing electricity supply and demand by regulating power consumption instead of generation. DR is a key technology for emerging smart electric grids that aim to increase grid efficiency, while incorporating significant amounts of clean renewable energy sources. In today's grid, DR is a rare event that only occurs when actual peak demands exceed the expected peak. In contrast, smart electric grids incentivize consumers to engage in continuous policy-driven DR to 1) optimize power consumption for time-of-use pricing and 2) deal with power variations from non-dispatchable renewable energy sources. While data centers are well-positioned to exploit DR, applications must cope with significant, frequent, and unpredictable changes in available power by regulating their energy footprint.

The problem is challenging since data centers often use distributed storage systems that co-locate computation and storage, and serve as a foundation for a variety of stateful distributed applications. As a result, existing approaches that deactivate servers as power decreases do not translate well to DR, since important application-level state may become completely unavailable. In this paper, we propose a DR-compatible storage system that uses staggered node blinking patterns combined with a balanced data layout and popularity-based replication to optimize I/O throughput, data availability, and energy-efficiency as power varies. Initial simulation results show the promise of our approach, which increases I/O throughput by at least 25% compared to an activation approach when adjusting to real-world wind and price fluctuations.

Categories and Subject Descriptors

C.5.0 [**Computer System Implementation**]: General

General Terms

Design, Management, Performance

1. INTRODUCTION

Data centers are rapidly expanding to accommodate the growth of online cloud-based services—facilities with hundreds of thousands of servers and up to a million cores are on the horizon. As is now well-known, larger data centers require more energy to power and cool their servers. The most recent estimates attribute 0.3% of *all* U.S. energy consumption specifically to data centers [11], with consumption estimated to double every 5 years [1]. Since over 83% of electrical energy in the U.S. derives from burning dirty fossil fuels, this rising energy demand has serious environmental ramifications [17]. Additionally, since the growth in society's energy demand is beginning to outpace its ability to locate, extract, and burn fossil fuels, energy prices are on a long-term upward trend [12]. Even using the "cheap" power available today, the energy-related costs of data centers already represent a significant fraction (~31% [10]) of their total cost of ownership. Thus, designing techniques to reduce the financial and environmental impact of data center energy consumption is an important problem. Energy price trends have already led data centers to experiment with alternative energy sources, such as wind [9], solar [20], and tidal [21].

Prior research focuses largely on increasing overall energy-efficiency by reducing data center energy consumption. For example, power-proportional data centers and servers that consume power in proportion to workload demands, e.g., by deactivating idle servers, increase efficiency by significantly reducing wasted power from idle servers [6, 26]. Likewise, balanced systems increase efficiency by reducing wasted power from idle server components [5, 23]. However, little prior research focuses on optimizing *when* data centers consume power. The timing of data center power consumption has a disproportionate affect on both the monetary cost and carbon footprint of power generation for at least two reasons.

Heterogenous Generators. Utilities operate a variety of generators that vary in their carbon emissions and fuel costs. For instance, the "peaking" generators that satisfy transient demand peaks have significantly higher emissions and fuel costs than the baseload generators that operate continuously. Thus, electricity generated during peak hours is more expensive and "dirty" than off-peak hours.

Intermittent Renewables. Both data centers and utilities are beginning to harvest energy from intermittent renewable power sources, such as wind and solar, which have no carbon footprint or fuel costs. As a result, data centers have an opportunity to decrease their carbon footprint by aligning when they consume power with when these clean energy sources are producing it.

The economics of power generation is also motivating utilities to adopt a variety of pricing structures that better reflect generation costs. As one example, utilities often add surcharges to electricity bills based on peak electricity usage over a billing cycle to incentivize consumers to "flatten" their electricity demand. As another example, many utilities are shifting from flat-rate pricing models for consumers to time-of-use pricing models that allow electricity prices to rise and fall with demand. Such market-based pricing, which is already common in wholesale electricity markets, incentivizes consumers to shift their usage to off-peak hours. Additionally, to further encourage the use of intermittent renewables, governments are beginning to impose cap-and-trade policies that artificially increase the cost of energy from fossil fuels. For example, a cap-and-trade policy went into effect in the U.K. in April 2010

GreenNet'11, August 19, 2011, Toronto, Ontario, Canada.

for businesses consuming more than 6GWh per year [14]. Each of the examples above represent strong monetary incentives for consumers, including data centers, to regulate not only how much power they consume, but also when they consume power. These incentives are an integral part of recent smart grid efforts.

Demand response (DR) is a general term for dynamically regulating energy consumption over time. In today's electric grid, utilities typically implement DR by manually signaling large consumers, e.g., via phone or email, to request reductions in their electricity usage during times of grid congestion or capacity constraints. Since the grid has the generation capacity to meet estimated peak demands, it uses DR rarely to prevent unexpected outages or "brownout" scenarios. As smart grid efforts expand, we envision consumers engaging in *continuous policy-driven DR* to optimize power consumption for time-of-use market-based pricing or to deal with power variations from non-dispatchable renewable energy sources. In this case, rather than the utility directing consumer DR on rare occasions, consumers will automatically decide when and how to consume power based on automated policies that precisely control their cost and carbon footprint.

Data centers are particularly well-positioned to adopt and benefit from continuous policy-driven DR for at least three reasons.

- First, servers already include sophisticated power management mechanisms that are remotely programmable. As a result, data centers are capable of programmatically varying their power consumption over a wide dynamic power range.
- Second, many workloads are tolerant to delays or performance degradation, enabling data centers to adjust their power consumption over time. The price elasticity of demand is higher for these flexible workloads than many household and industrial loads, which are not highly responsive to price fluctuations.
- Finally, data centers are large industrial power consumers that have a substantial impact on grid conditions—utilities typically target these large consumers for DR first.

Due to the explosive growth of data center size and energy consumption, a 2007 EPA report to Congress on data center energy efficiency encourages them to adopt DR to reduce their strain on the electric grid [1]. However, *to exploit DR, the key challenge that systems researchers must address is designing applications that perform well in the face of significant, frequent, and potentially unpredictable variations in power.* Thus, designing for DR differs significantly from past research efforts on designing for energy-efficiency or proportionality. Rather than optimizing the energy required to satisfy a given workload, designing for DR requires systems to optimize application performance as power varies. The problem is particularly challenging *since power fluctuations may occur independently of workload demands.*

2. BLINKING: A GENERAL DR ABSTRACTION FOR DATA CENTERS

There are multiple possible approaches to handling power variations in data centers. One option is to migrate workload, in the form of virtual machines (VMs), from locations with power shortages to those with power surpluses [3]. For example, solar-powered data centers may migrate VMs to daytime locations to handle nighttime power shortages. Power costs for geographically-disparate data centers may differ for a variety of other reasons as well [22], motivating migrations to data centers with cheap power. However, despite recent advances in optimizing live VM migration for the wide-area network links between data centers [28], the approach is not suitable for data-intensive workloads that access vast amounts of storage. Even with dedicated high-bandwidth network links, daily transfers of multi-petabyte storage volumes is infeasible. Further, while solar power may vary somewhat slowly and predictably on a daily basis, in general, data centers may wish to optimize for power variations that are much more rapid and unpredictable, e.g., from changing spot prices or wind power.

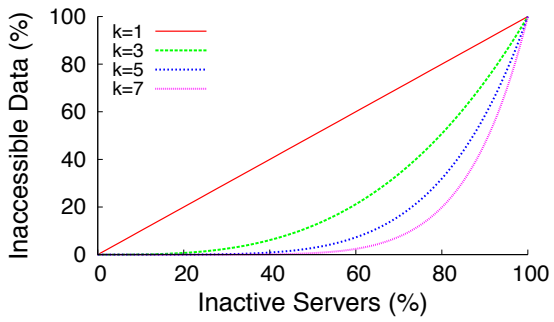

Figure 1: Inactive servers result in inaccessible data. Replication mitigates, but does not eliminate, the problem, especially for high percentages (>50%) of inactive servers.

We recently proposed a new technique, called blinking [24], to regulate a data center's energy footprint in response to power variations. Blinking rapidly transitions servers between a high-power active state and a low-power inactive sleep state. Since power consumption in the inactive state, e.g., ACPI's S3 state, is minimal, it is possible to regulate cluster energy consumption over small time scales by changing the relative duration of each server's active and inactive modes. We briefly outline our implementation of a prototype blinking cluster in Section 4. In prior work, we demonstrate how to enable blinking in memcached, a stateless distributed memory cache [24]. However, we intend blinking to be a *general abstraction for implementing DR for a wide range of data center applications*. In this paper, we examine how blinking might enable DR for the distributed storage systems now common in data centers. Since power variations result in servers becoming periodically unavailable, distributed storage represents one of the most challenging problems for implementing DR in data centers.

The blinking abstraction supports a variety of *blinking policies*. For instance, an *activation* policy is any blinking policy that only varies the fraction of inactive servers to vary power consumption. Activation policies are commonly used to ensure data centers are energy-proportional by varying the fraction of active servers to match demand [4, 16, 26]. Prior research has also used activation policies to address power variations from renewable energy sources, although the primary focus is on simple computationally-intensive workloads using always-accessible storage [15]. The blinking abstraction permits other more sophisticated policies as well, including *synchronous* policies that transition servers between the active and inactive states in tandem, *asynchronous* policies that stagger server active periods over time, and various *asymmetric* policies that blink servers at different rates based on application-specific performance metrics. In the next section, we focus on issues with enabling DR for distributed storage, and discuss the disadvantages of commonly-used activation policies.

3. PROBLEMS WITH ACTIVATION

Distributed storage layers, such as HDFS [25], distribute data across compute servers in data centers, and serve as the founda-

tion for many higher-level distributed data center applications. An activation policy that naïvely deactivates servers to save energy or match a given level of available power is capable of making the data on inactive servers inaccessible. One way to prevent data on inactive servers from becoming inaccessible is by storing replicas on active servers. Replication is already used to improve the performance and reliability of distributed storage, and works well for an activation policy if the number of inactive servers is small. For example, with HDFS's random replica placement policy, the probability that any block is unavailable is $\frac{m!(n-k)!}{n!(m-k)!}$ for n servers, m inactive servers, and k replicas per block. Figure 1 plots the expected percentage of inaccessible data as a function of the percentage of inactive servers, and shows that nearly all data is accessible for small percentages of inactive servers.

However, the amount of inaccessible data rises dramatically once 50% of servers are inactive, even for aggressive replication factors, e.g., $k = 7$. Frequent periods of low power and many inactive servers is a common case for many DR scenarios. Further, even a few inactive servers, where the expected percentage of inaccessible data is small, have the potential to cause significant performance issues, e.g., by stalling large batch jobs that are dependent on some inaccessible data. One way to address the problem is through careful placement of data and replicas. For instance, prior work on energy-efficient distributed storage proposes data layouts that enable servers to be deactivated without causing any data to become unavailable [4, 13, 18]. In general, the proposed layouts store primary replicas on one subset of servers, secondary replicas on another mutually-exclusive subset, tertiary replicas on yet another subset, etc. However, these *concentrated data layouts* introduce at least three issues if available power varies frequently and significantly, and is independent of workload demands.

Wasteful Thrashing. Power variations may trigger frequent changes to the number of the active servers storing primary, secondary, tertiary, etc. replicas that require migrations to either 1) spread data out to provide higher I/O throughput or 2) concentrate data to keep it accessible. Minimizing these migration overheads is a focus of prior work on energy-efficient storage [29].

Wasted Capacity. Recent work on energy-proportional distributed storage reduces migration-related thrashing by making replica sets progressively larger, e.g., more servers store secondary replicas than primary replicas [4]. Assuming servers have similar storage capacity, the approach wastes progressively more capacity on the servers storing secondary, tertiary, etc. replicas.

Inaccessible Data. Regardless of the data layout, if there is not enough power to activate the necessary servers to store a complete data set, then some data will be inaccessible. Thus, even data layouts that ensure a small "covering" subset of servers store all primary replicas has the potential to make data inaccessible [18],

To highlight the problems of wasteful thrashing and inaccessible data, consider a simple example where there is enough power to currently operate $2N$ servers storing a data set's primary replicas. Also assume that the data set fills the storage capacity of N servers. Now consider the consequences of a sudden and unexpected drop in power by a factor 2, which leaves only N servers active. To ensure that the data is available, we must migrate to a new concentrated data layout that spreads data across the N active servers. However, if we did not expect the drop, there may not have been enough time to migrate to the more concentrated data layout. In this case, 50% of the data may be inaccessible even if we have enough capacity to store it on N active servers. Now consider what happens if available power drops again by a factor of 2, which leaves only $0.5N$ remaining active servers. Since the data set fills the storage capacity of N servers, migration is not even possible and 50% of the data set is inaccessible at best. Figure 2 graphically depicts our example for $N = 4$. Note that power increases also require time-consuming data migrations to spread data out and increase aggregate I/O throughput. In the next section we discuss a blinking approach that addresses the problems above, while providing better I/O throughput, data availability, and energy-efficiency.

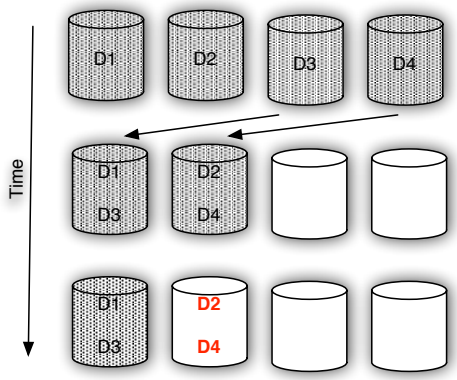

Figure 2: Power variations may cause the subset of active servers storing a data set's primary replicas to decrease, which triggers costly migrations to keep data accessible. At low power levels, some data is certain to be inaccessible.

4. BLINKING DISTRIBUTED STORAGE

We first provide a brief summary of blinking. Our prior work includes a complete description, and applies the concept to a simple distributed memory cache that does not persist data [24]. Blinking is a simple mechanism that enables an external controller to remotely set a blink interval t and an active interval t_{active} on a server, such that for every interval t a server is active for time t_{active} and inactive for time $t - t_{active}$. ACPI's S3 state is a good choice for the inactive state, since it combines the capability for fast millisecond-scale transitions with low power consumption (<5% peak power). In contrast, component-level techniques, such as DVFS, are much less effective at satisfying steep drops in available power, since they are often unable to reduce consumption below 50% peak power [27]. To control inter-server blinking patterns, the blinking abstraction also enables a controller to specify when a blink interval starts, as well as when within a blink interval the active interval starts.

To illustrate the advantages of blinking distributed storage for DR, we consider our example from the previous section and discuss how blinking affects I/O throughput, data availability and request latency, and energy-efficiency. We assume the use of flash-based SSDs, which have already been shown to be more energy-efficient than disks for a range of both seek- and scan-intensive workloads [5, 23, 27]. As a result, flash-based SSDs are becoming increasingly popular for data-intensive workloads. Flash-based SSDs are also compatible with blinking, which relies on frequent and rapid transitions between power states. Blinking could also be applicable to mechanical disks, but would require a new low-power state that is similar to ACPI's S3 state but permits disks to remain spun-up. Frequent disk spin-up and spin-down transitions have been shown to degrade their reliability [29].

4.1 I/O Throughput

Recall that in our example there is initially enough power to operate $2N$ servers that each provide storage for a fraction of our

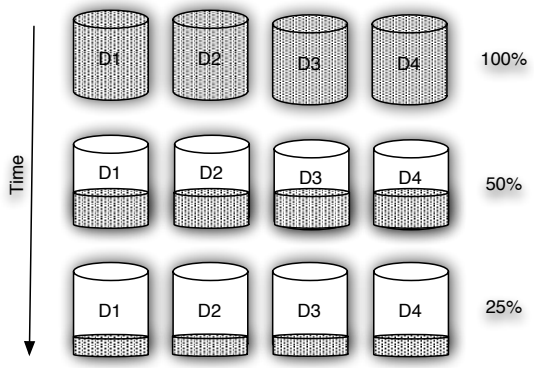

Figure 3: Power variations decrease the fraction of time each server is active during a blink interval. No migrations are necessary to adjust to power variations, since all blocks are accessible for a fraction of each blink interval.

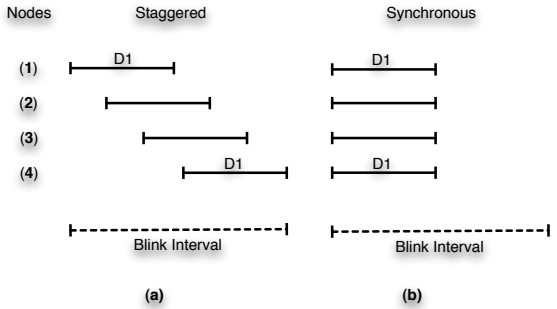

Figure 4: A staggered server blinking pattern is able to use replication to maximize the availability of data and decrease average I/O latency as power varies.

data set. If the available power decreases by a factor of 2, with blinking we have the option of keeping $2N$ servers active for time $t_{active} = \frac{t}{2}$ every blink interval t. In this case, instead of migrating data and concentrating it on N active servers, we are able to keep the same data layout as before without changing our maximum I/O throughput over each blink interval, assuming negligible transition overheads. To see why, consider that a layout that concentrates data storage on N servers is capable of NM MB/sec, assuming a maximum I/O throughput per server of M MB/sec, while a distributed layout on $2N$ servers with blinking is capable of $2NM$ MB/sec for half of every blink interval (or NM MB/sec on average).

Our example highlights that at any fixed power level, blinking is able to provide the same maximum I/O throughput, assuming negligible transition overheads, as an activation approach. Blinking, however, has a distinct advantage over activation if the available power changes, since it is possible to alter server active intervals nearly instantly to match the available power. In contrast to an activation approach, a blinking approach need not waste time and bandwidth by migrating data to 1) keep it accessible or 2) increase the maximum I/O throughput. If power suddenly drops again by another factor of 2, blinking is able to nearly instantly reduce each server's active interval by another 2x ($t_{active} = \frac{t}{4}$). Whereas with activation 50% of the data is completely inaccessible after the drop, blinking gracefully handles the drop by ensuring that 100% of the data is accessible for 25% of every blink interval. The average drop in aggregate I/O throughput per block is also proportional to the power drop (a factor of 2 to $0.5NM$ MB/sec). Since no migrations are necessary to gracefully handle the sudden power drop, there is no risk of inaccessible data due to unfinished migrations. Figure 3 graphically depicts our simple example with $N = 4$.

4.2 Data Availability and Latency

Blinking's primary drawback is a significant increase in average latency to retrieve data, since any request for data to an inactive server stalls until the server becomes active. If we assume that data access latency is negligible for data stored on active servers, then average latency increases from near 0 with a concentrated data layout in an activation approach (assuming all data is accessible) to $\frac{t-t_{active}}{2}$ with blinking, assuming request interarrival times are randomly distributed. Since blink intervals are on the order of 10s of seconds for platforms with transition latencies of a few hundred milliseconds, the latency penalty of our naïve blinking policy is high. While a long average request latency is not important for long-running batch jobs that take many blink intervals to complete, it is important for shorter jobs or more interactive workloads.

One way to increase data availability and reduce average I/O latency with blinking is using replication. Staggering server active intervals equally over each blink interval, by 1) ordering servers n_i for $i = 1, 2, 3, ...N$, 2) starting their active intervals in order, and 3) pausing for time t_{pause} after each activation, enables a simple replica placement policy that maximizes a block's availability over each blink interval, regardless of the available power. In this case, the size of t_{pause} varies according to available power. The placement policy spaces out block replicas among servers with active intervals that start as far apart as possible within each blink interval to minimize overlap in the active intervals. We contrast a staggered blinking schedule with a synchronous blinking schedule in Figure 4. While both blinking schedules provide the same I/O throughput, our staggered schedule reduces the average latency to access $D1$ by ensuring it is available for the entire blink interval.

As we discuss in our preliminary analysis, significantly reducing average latency may require aggressive replication. However, aggressively replicating all blocks significantly reduces storage capacity. Thus, *popularity-based replication and reclamation* of individual blocks is an important optimization for maintaining both low access latency and ample storage capacity. In this case, as blocks become more popular more replicas are deployed to increase availability, and as blocks become less popular more replicas are reclaimed to increase capacity. Additionally, data centers could use small amounts of stored power from battery arrays to power additional servers and accelerate replication for blocks that gain popularity rapidly under low-power situations. While recent work demonstrates how to leverage stored power for multi-tier web applications [8], DR for distributed storage represents another opportunity for leveraging local energy storage.

Further, unlike data layouts that concentrate replicas on mutually-exclusive subsets of servers, blinking enables replica placement of blocks on servers to *balance* I/O throughput among servers. Load balancing to ensure each server stores equally popular data is a worthy goal at both high power—with all servers active—and low power—with a staggered blinking pattern. In contrast, the appropriate size of a server subset in a concentrated data layout changes with available power. One advantage of blinking is that data layout decisions, which are costly to change, need not be based on the available power level: replication combined with equally-sized, staggered active intervals reduces average latency under low power, while improving I/O throughput and reliability

Figure 5: Wind turbine and solar panel deployment on the roof of the Computer Science Building at the University of Massachusetts Amherst.

under high power. Finally, recent work [19] has demonstrated the difficulty of making online data-intensive services energy-proportional while maintaining low latency data access using conventional per-server power modes. Our DR-motivated techniques may also prove useful in this context.

4.3 Energy-efficiency

Blinking also improves energy-efficiency. Recent research points out the contradiction between deactivating servers to save energy and load balancing to decrease cooling costs, since fully active servers introduce hotspots that increase the energy consumption of cooling [2]. In contrast, staggering active intervals saves the same energy as deactivating servers, while eliminating hotspots by load balancing heat generation across all servers. Balancing power equally across servers is also useful under 3-phase power, since unequal consumption across phases wastes power and generates heat that requires additional energy to cool. Finally, since a staggered blinking policy minimizes the set of servers that is concurrently active, it reduces peak power. Both operational and capital costs are disproportionately affected by peak, rather than average, power [7].

5. PRELIMINARY ANALYSIS

To gain insight into DR's potential benefits for data centers, we simulated two DR scenarios for a small-scale 15-server prototype we are building, which uses SSD-enabled Mac minis connected to programmable power supplies that drive variable power traces. While DR is applicable in many scenarios, our analysis focuses on 1) powering our prototype using wind energy and 2) adhering to a fixed power budget despite variable energy prices. Both scenarios generate a variable supply of available power. For wind energy, we scale a power signal collected from our own small-scale wind turbine deployment [24], such that its average power is equal to the power our cluster requires. Figure 5 is a photograph of our wind turbine and solar panel deployment on the roof of the Computer Science Building. Since wind often generates no power for long time periods, we assume a base power level of 5% peak cluster power (18.75W). For variable prices, we use the New England ISO's 5-minute spot price of energy for the last 3 months of 2010.

Each Mac mini consumes a maximum of 25W for a total cluster power consumption of 375W. For our simulation, we assume each SSD stores 10GB of data and has infinite capacity to prevent inaccessible data due to a lack of storage space. Since the S3 transition latency is roughly 1 second, we assume a 60 second blink interval.

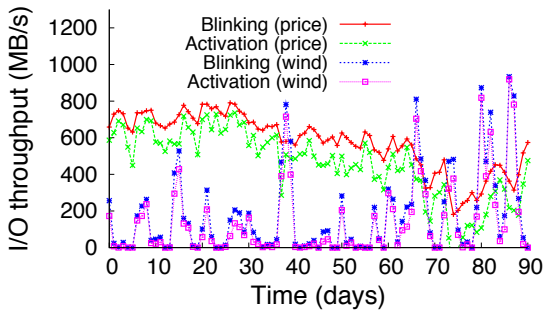

Figure 6: The maximum aggregate I/O throughput for blinking is greater than for activation for both our wind energy and variable price scenarios.

Our analysis assumes a maximum I/O throughput of 65 MB/sec per server, which represents roughly an equal mix of (fast) reads and (slow) writes to the SSD. We also assume that the migration speed between servers is equal to the maximum per-server I/O throughput. We then compare the maximum I/O throughput possible over a 3-month period for activation and blinking in both scenarios, where the activation approach must finish migrating data before accessing it and the blinking approach incurs 1 second of downtime every 60 seconds. The maximum aggregate I/O throughput is one important measure of the performance of a distributed storage system.

Figure 6 shows that blinking's I/O throughput is higher than activation in both scenarios. For wind, the average I/O throughput over the 3 month period is 174 MB/sec for blinking and 136 MB/sec for activation, yielding an improvement of nearly 28%. For variable prices, we use a budget of 2.67¢/kWh, which for our small-scale cluster, which translates to paying 1¢/hour for energy over the 3-month period. By comparison, the average price in our trace is 4.55¢/kWh, the peak price is 33.16¢/kWh, and the minimum price is 0.97¢/kWh. In this case, the average I/O throughput over the 3 month period is 575 MB/sec with blinking and 457 MB/sec with activation, yielding an improvement of over 25%. Thus, blinking enables either 25% more work to be done for the same budget, or the same work to be done for 25% less money.

Figure 7 shows that replication reduces average latency with blinking significantly, assuming a 20ms latency to access data on active servers. Since wind requires aggressive replication (10x) to reduce average latency below 1 second, due to many periods of no power, per-block popularity-based replication is an important optimization. Note that we are not advocating permanent 10x replication for every data object to decrease the latency penalty of intermittent power. Rather, we propose aggressive per-object popularity-based replication as objects become more popular, and aggressive replica reclamation as objects become less popular. Our premise is that the average latency for data-intensive workloads that exhibit high degrees of temporal locality should decrease significantly with per-object popularity-based replication, without a significant decrease in the aggregate storage capacity.

We view our simulation results, while admittedly rough, as a conservative estimate of blinking's benefits. In reality, network transfers will likely not be as fast as I/O, especially since the cluster will be using network bandwidth to do useful work, which will reduce activation's I/O throughput. Additionally, we assume only enough energy storage for 60 seconds. For wind energy, there are periods where our cluster is unable to consume 100% of the available wind energy. Thus, slightly more energy storage could provide significant benefits. Interestingly, with renewables, it is possible to

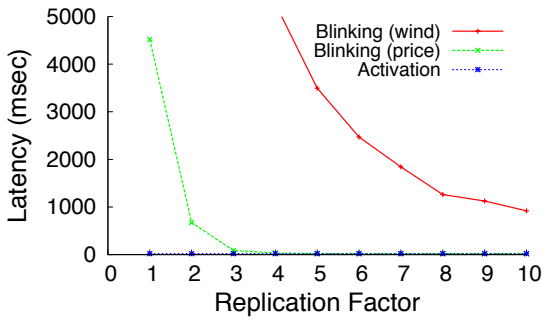

Figure 7: A small amount of replication is capable of significantly improving data access latency with blinking.

waste energy by not consuming more of it. We also do not quantify any of the potential indirect benefits to energy-efficiency discussed in Section 4.3. Finally, we assume data is never inaccessible with activation, which is not likely for significant power fluctuations.

6. CONCLUSION AND FUTURE WORK

While we focus on two examples—renewable energy and variable pricing—DR is applicable in other scenarios. For instance, DR is useful during grid power outages or when data center PDUs fail. Another compelling example is continuously running background workloads that always consume the excess power PDUs are capable of delivering. Since data center capital costs are enormous, maximizing the power delivery infrastructure's utilization by operating as many servers as possible is important. However, data centers provision power for peak demands, resulting in low utilization [7].

We are currently designing a distributed storage layer for DR. There are numerous challenges to address in a real system, although space limitations afford only a brief summary of them here. We are studying specific policies for deciding when and where to spawn or destroy replicas based on data popularity and application-specific performance goals. Additionally, applications that co-locate computation and data require a new mechanism to transfer data between servers if they are never concurrently active. Another challenge is ensuring consistency between replicas, while maintaining high throughput. In the worst case, an entire blink interval is necessary to ensure consistency. Performing well for write-intensive workloads is related, since writes apply to replicas on both active and inactive servers. We are exploring using a small set of always-active proxies to address these issues. Proxies absorb writes for an entire blink interval, and then apply them in order as servers become active in the next interval. Proxies may also act as routers, to transfer data between servers that are not concurrently active, or caches to store newly popular data during the replication process.

In conclusion, in this paper, we motivate DR for data centers, discuss the issues with enabling DR for distributed storage, and propose a DR-compatible blinking-based approach. While more work is necessary to validate our ideas, given current trends in energy consumption and pricing, we believe that DR is worth exploring in future data centers.

Acknowledgements. This work was supported by the National Science Foundation under grants CNS-0855128, CNS-0834243, CNS-0916577, and EEC-0313747.

7. REFERENCES

[1] U.S. Environmental Protection Agency. Report to Congress on Server and Data Center Energy Efficiency. August 2007.

[2] F. Ahmad and T. Vijaykumar. Joint Optimization of Idle and Cooling Power in Data Centers while Maintaining Response Time. In *ASPLOS*, March 2010.

[3] S. Akoush, R. Sohan, A. Rice, A. Moore, and A. Hopper. Free Lunch: Exploiting Renewable Energy for Computing. In *HotOS*, May 2011.

[4] H. Amur, J. Cipar, V. Gupta, M. Kozuch, G. Ganger, and K. Schwan. Robust and Flexible Power-Proportional Storage. In *SoCC*, June 2010.

[5] D. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan. FAWN: A Fast Array of Wimpy Nodes. In *SOSP*, October 2009.

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

[7] X. Fan, W. Weber, and L. Barroso. Power Provisioning for a Warehouse-Sized Computer. In *ISCA*, June 2007.

[8] S. Govindan, A. Sivasubramaniam, and B. Urgaonkar. Benefits and Limitations of Tapping into Stored Energy for Datacenters. In *ISCA*, June 2011.

[9] P. Gupta. Google to use Wind Energy to Power Data Centers. In *New York Times*, July 20th 2010.

[10] J. Hamilton. Overall Data Center Costs. In *Perspectives. http://perspectives.mvdirona.com/*, September 18, 2010.

[11] J. Hamilton. 2011 European Data Center Summit. http://perspectives.mvdirona.com/, May 25, 2011.

[12] R. Hirsch, R. Bezdek, and R. Wendling. Peaking of World Oil Production: Impacts, Mitigation, and Risk Management. In *U.S. Department of Energy*, February 2005.

[13] R. Kaushik and M. Bhandarkar. GreenHDFS: Towards an Energy-Conserving Storage-Efficient, Hybrid Hadoop Compute Cluster. In *USENIX Annual Technical Conference*, June 2010.

[14] United Kingdom. CRC Energy Efficiency Scheme. http://www.decc.gov.uk/, June 2011.

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

[16] A. Krioukov, P. Mohan, S. Alspaugh, L. Keys, D. Culler, and R. Katz. NapSAC: Design and Implementation of a Power-Proportional Web Cluster. In *GreenNet*, August 2010.

[17] Lawrence Livermore National Laboratory. U.S. Energy Flowchart 2008. https://flowcharts.llnl.gov/, June 2011.

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

[19] D. Meisner, C. Sadler, L. Barroso, W. Weber, and T. Wenisch. Power Management of Online Data-Intensive Services. In *ISCA*, June 2011.

[20] R. Miller. Microsoft to use Solar Panels in New Data Center. In *Data Center Knowledge*, September 24th 2008.

[21] R. Miller. Morgan Stanley Plans Tide-Powered Data Center. In *Data Center Knowledge*, October 17th 2008.

[22] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs. Cutting the Electric Bill for Internet-Scale Systems. In *SIGCOMM*, August 2009.

[23] S. Rivoire, M. Shah, , and P. Ranganathan. JouleSort: A Balanced Energy-Efficient Benchmark. In *SIGMOD*, June 2007.

[24] N. Sharma, S. Barker, D. Irwin, and P. Shenoy. Blink: Managing Server Clusters on Intermittent Power. In *ASPLOS*, March 2011.

[25] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The Hadoop Distributed File System. In *MSST*, May 2010.

[26] N. Tolia, Z. Wang, M. Marwah, C. Bash, P. Ranganathan, and X. Zhu. Delivering Energy Proportionality with Non-Energy-Proportional Systems: Optimizing the Ensemble. In *HotPower*, December 2008.

[27] V. Vasudevan, D. Andersen, M. Kaminsky, L. Tan, J. Franklin, and I. Moraru. Energy-efficient Cluster Computing with FAWN: Workloads and Implications. In *e-Energy*, April 2010.

[28] T. Wood, K. Ramakrishnan, P. Shenoy, and J. Van der Merwe. CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines. In *VEE*, March 2011.

[29] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes. Hibernator: Helping Disk Arrays Sleep Through the Winter. In *SOSP*, October 2005.