Underprovisioning Backup Power Infrastructure for Datacenters

Di Wang* Sriram Govindan† Anand Sivasubramaniam* Aman Kansal‡
Jie Liu ‡ Badriddine Khessib†

*The Pennsylvania State University, †Microsoft Corporation, ‡Microsoft Research
{diw5108, anand}@cse.psu.edu, {srgovin, kansal, jie.liu, bkhessib}@microsoft.com

Abstract

While there has been prior work to underprovision the power distribution infrastructure for a datacenter to save costs, the ability to underprovision the backup power infrastructure, which contributes significantly to capital costs, is little explored. There are two main components in the backup infrastructure - Diesel Generators (DGs) and UPS units - which can both be underprovisioned (or even removed) in terms of their power and/or energy capacities. However, embarking on such underprovisioning mandates studying several ramifications - the resulting cost savings, the lower availability, and the performance and state loss consequences on individual applications - concurrently. This paper presents the first such study, considering cost, availability, performance and application consequences of underprovisioning the backup power infrastructure. We present a framework to quantify the cost of backup capacity that is provisioned, and implement techniques leveraging existing software and hardware mechanisms to provide as seamless an operation as possible for an application within the provisioned backup capacity during a power outage. We evaluate the cost-performance-availability trade-offs for different levels of backup underprovisioning for applications with diverse reliance on the backup infrastructure. Our results show that one may be able to completely do away with DGs, compensating for it with additional UPS energy capacities, to significantly cut costs and still be able to handle power outages lasting as high as 40 minutes (which constitute bulk of the outages). Further, we can push the limits of outage duration that can be handled in a cost-effective manner, if applications are willing to tolerate degraded performance during the outage. Our evaluations also show that different applications react differently to the outage handling mechanisms, and that the efficacy of the mechanisms is sensitive to the outage duration. The insights from this paper can spur new opportunities for future work on backup power infrastructure optimization.

Categories and Subject Descriptors C.0 [*Computer Systems Organization*]: General

Keywords Datacenters, Backup Power Infrastructure, Underprovision, UPS, Diesal Generator

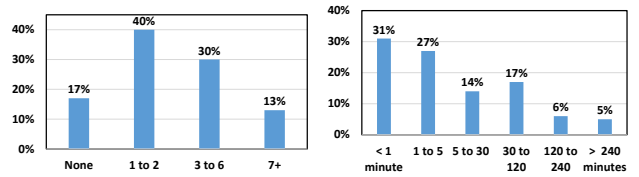
ASPLOS '14, March 1–4, 2014, Salt Lake City, Utah, USA.

http://dx.doi.org/10.1145/2541940.2541966

1. Introduction

Considerable capital cost is spent on provisioning the power infrastructure for a datacenter. It can cost between $10-$25 for each watt of power provisioned [11, 29, 30, 34], even if that watt is not consumed, contributing to over a third of the amortized costs of a large datacenter. Consequently, recent works [22, 23, 29, 34, 63] have proposed to underprovision this infrastructure, sizing it to handle a high percentile of the power draw rather than the occasional high peaks. Power capping mechanisms are then employed to ensure safety when this limit is reached. Aside from the cost of equipment needed for power distribution, conversion and quality (e.g. switchgear, transformers, PDUs, etc.), a significant portion of this infrastructure's capital cost (over 20% [45]) goes into the backup power equipment - Diesel Generators (DGs) and Uninterrupted Power Supplies (UPS) - needed to sustain operation during utility power outages. To our knowledge, while much prior work has focused on underprovisioning the overall power distribution infrastructure in datacenters, the cost-benefit trade-offs of under-provisioning the backup capacity have not been considered. In this paper, we investigate the relatively unexplored area of underprovisioning, and perhaps even removing, parts of the backup power infrastructure.

Analogous to the distribution infrastructure which needs to handle occasional high peak draws, the backup infrastructure is called upon only when there is an occasional power outage. Figure 1 shows the distribution of power outages in a typical year for US businesses [50, 60]. In Figure 1(a), we observe that 6 or fewer outages are the overwhelming majority (in 87% of the businesses). Further, in Figure 1(b) we see that a large majority (over 58%) of these outages are



(a) Power Outage Frequency per Year (b) Power Outage Duration

Figure 1. Power Outages Distribution for U.S. Business (Source: [50, 60])

shorter than 5 minutes. While handling these rare and typically short outages is extremely important for a datacenter's availability, it is not clear whether the associated high costs for provisioning the backup infrastructure for every eventuality is justified. Provisioning a cost-effective backup infrastructure for datacenters based on these outage characteristics poses interesting research challenges. *Should we provide full backup capacity for the very long (> 4 hours) but extremely rare power outages?* With geo-replication often in use across large datacenters of an organization, a rare and prolonged outage may possibly be handled by load redirection/migration to other (power uncorrelated) sites, even if it is at a slight performance overhead in order to save backup infrastructure costs (e.g. as discussed in [32]). At the other end, *how should we provision a cost-effective backup to handle the load during periods of relatively shorter outages, which could be more frequent?* This paper investigates these questions, with more emphasis on the latter, while considering the consequences of cost-effective backup infrastructure choices on the resulting availability and performance of the datacenter.

Underprovisioning Backup and Its Consequences: Backup capacity costs can be expressed in two dimensions - power and energy, i.e. power load that needs to be sustained during the outage, and the energy (integral of power over time) in this period. These impact the cost of provisioning the two main components of the backup infrastructure, namely the Diesel Generators (DGs) and the UPS units. A DG's capital cost is mainly determined by its peak power load, much more than the energy (the cost of building fuel tanks whose capacity impacts the latter is much smaller than the DG itself). On the other hand, UPS cost is determined both by the power load as well as the duration (energy) over which its associated batteries have to be employed. In today's datacenters, UPSes are used mainly as a transition mechanism to switch over the load to Diesel Generators, which can take several seconds to a few minutes, to provide seamless operation for the computing load. However, there is no reason why one could not provision extra battery energy over and beyond this need, if it can offset some or all of the costs associated with DG provisioning. For instance, if we are to only handle outages lasting up to 10 minutes, it may be more cost-effective to eliminate DGs altogether and provision extra battery capacity to last 10 minutes.

When underprovisioning the backup infrastructure, we can have a spectrum of choices between (i) current practice of having full UPS power + energy capacity to hand over the load to DGs, which then can sustain the power need for the entire duration of the outage (assuming sufficient fuel reserve), to (ii) having no UPS or DGs, where the datacenter cannot operate during the outage. Between these extremes, one could opt for different operating points including (a) varying DG power capacity, (b) varying UPS power capacity, (c) varying UPS energy, or (d) combinations thereof. Based on such choices, compute availability and performance could be reduced during the power outage because some or all of the servers are powered off or operating in a lower power state. With reduced backup capacity, application state may be lost in case of an outage if the servers lose power abruptly, before the state is persisted on a stable medium. Within a provisioned backup power and energy capacity, there are different alternatives to improving such performance and availability, such as consolidation and shutting down servers, using low power working states, saving application state etc. In this paper, we use the term performability to loosely refer to both performance and availability of the datacenter during (and after) a power outage, though we quantify the two metrics (performance and availability) separately in our evaluation.

Overview and Contributions: This paper explores the design space trade-offs when building different backup capacity alternatives for a datacenter. Within the confines of a provisioned backup capacity, we leverage software and hardware techniques to enhance the performability of the datacenter during power outages of different duration. Some specific issues that we investigate include: Can we completely do away with DGs? Up to what outage duration? What is the minimum cost, and the resulting backup capacity, to handle different outage durations? What are the consequent ramifications on performability? How sensitive are different kinds of applications to these issues? What system mechanisms and online strategies are needed to improve the performability of these applications within the stipulated capacities?

The following are some of the insights from our evaluation of a spectrum of workloads with diverse dependence on the backup infrastructure:

- For outages up to 40 mins, DGs are not needed. It is cost effective to buy additional UPS energy capacity to support the availability and performance mandates. While a DG does translate long outages to smaller ones from the perspective of offered performability, it does so at a significant cost. Given that an overwhelming percentage of power outages are of the order of at most a few minutes, this warrants a re-thinking of current infrastructures which use DGs.
- UPSes play a crucial role in outages of short duration, and can be the sole backup for outages up to 100 minutes to offer similar performability at a similar cost as to-

day's infrastructures. If applications can tolerate a 40% performance degradation during such long power outages, we can get a 40% cost savings as well with just UPS as the backup source. Accommodating longer runtimes on a UPS battery is a more cost and performability effective alternative than using it for high power.

- Different applications react differently to the system mechanisms employed during an outage. In other words, the choice of the system mechanism for a given performability target can be different across multiple applications.
- In general, active power state modulation is better for short outages, sleep or hibernation along with power state modulation for medium outages, and migration and consolidation for long outages, for typical cost-performability tradeoffs.

2. Related Work

Datacenter Power Reduction/Under-provisioning: The high cost of power provisioning and consumption in datacenters has spurred several efforts to underprovision the power infrastructure [10, 14, 22, 23, 29, 34, 63, 65]. These efforts rely on system techniques for reducing demand and can be categorized in to (a) server-level power reduction using active processor power states [13, 20, 24, 36, 37, 49, 53, 54, 66, 67], inactive deep sleep states [1, 5, 6, 18, 41–43, 55], and power-aware workload scheduling techniques [17, 47], (b) consolidation via migration or load-distribution [4, 16, 39, 52, 59, 62], and the more recent (c) energy-storage/battery based peak shaving approaches [9, 27, 29, 34, 63].

Though related, *backup* power infrastructure underprovisioning and *normal* power under-provisioning are quite different: (a) Unlike normal distribution where only peak power capacity is under-provisioned, in the case of backup, both peak power as well as energy capacity could be underprovisioned. This is particularly true when we remove DGs, the potentially infinite energy source during an outage, creating several other considerations when operating within this limited capacity during an outage. (b) This limited energy capacity calls for additional system software mechanisms to handle the limited energy and the limited power capacity, as opposed to handling only limited power capacity (in normal under-provisioning). Prolonging operation becomes as important as capping peaks. (c) Backup needs to take over the entire load during outage, as opposed to complementing utility power in normal under-provisioning. This heightens the criticality of employed mechanisms to ensure seamless operation. (d) On the positive side, Backup is called upon only for occasional power outages, unlike peak suppression for normal power under-provisioning that could be more frequent. Consequently, issues such as battery wear due to rare outages are less important, and the need to ensure availability is more essential as explored in this paper.

Backup Infrastructure Costs: Specific solutions in this area have mainly studied varying the redundancy and placement configurations [28, 40] of the backup equipment, to derive different availability-cost options, popularized by the famous *Tier* [61] classification of datacenters.

Application State Maintenance: The loss of application state due to different kinds of faults such as server crashes, network failures, storage failures, and software bugs, has been extensively studied in prior work. Many of these solutions are relevant and applicable to power outages as well. In general, techniques such as state-machine replication, proactive check-pointing and logging can be used (e.g. [15, 19, 33, 51]) to save or replicate application state and resume from the saved state. Intermittently available power has been explicitly considered in [56], which uses proactive migration to save state on a remote server. Further, non-volatility in the memory hierarchy, whether it be through new memory technologies or with commercially available solutions such as NVDIMM [2], can maintain application state upon power failures without any backup power [48].

While we leverage many of the above solutions for reducing the peak power and energy requirement of the backup infrastructure, our contributions are in evaluating cost-performability trade-offs with different backup configurations, system mechanisms used during the outage, different outage durations, and diverse application characteristics.

3. Backup Infrastructure Cost Analysis

Power Hierarchy and Backup Infrastructure: Power enters the datacenter from the utility substation (Figure 2). Datacenters typically use a backup secondary power source such as Diesel Generators (DG) to handle utility outages [1] [2]. An Automatic Transfer Switch (ATS) detects primary utility failures and subsequently switches the load over to DGs. This transfer is not instantaneous and can take several seconds. To ensure seamless operation despite such delays, datacenters employ Uninterrupted Power Supply (UPS) units with batteries as a ride-through mechanism to facilitate the transfer. DGs and UPSes, thus, constitute the backup infrastructure (cost of ATS is relatively small and we do not consider it in this paper).

Figure 2 shows UPS units placed at the rack-level which is popular in today's datacenters (as in Facebook [21] and Microsoft [46]) due to its efficiency and cost advantage over conventional centralized placement. The UPS units can either be configured as *online* (in series) or *offline* (in paral-

[1] Access to multiple independent, multi-megawatt utility lines in the same location is very rare and therefore we consider only single utility connection to a datacenter in this paper.

[2] DGs are also used for planned maintainence of the power distribution network without disrupting service. However, we only focus on its provisioning for handling power outages in this paper. One could possibly leverage mobile substations [12] with rapid on-demand deployment (with 12 to 24 hour advance notice) for maintenance purposes in case of non-/underprovisioned DGs.

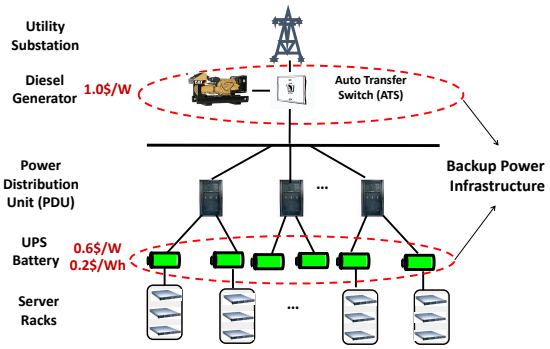


Figure 2. Datacenter Power Infrastructure

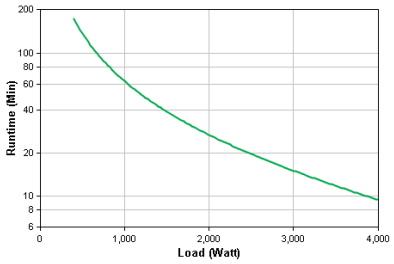


Figure 3. Runtime for a battery with max. power of 4KW.

lel), where the latter is preferred in today's datacenters to avoid double-conversion inefficiencies [38] associated with online UPSes. Unlike *online* UPSes which seamlessly transfer to sourcing from their batteries upon a utility failure, *offline* UPS design incurs a delay of about ˜10ms to detect a utility failure event before switching over. Fortunately today's power supplies have inherent capacitance to power the server for over 30ms to ride-through this transfer delay after a power failure [48]. It is important to note that *offline* UPS units are exercised only during power outage duration [3] and therefore any under-provisioning in UPS capacity is unlikely to impact normal operation (when utility is active).

The peak power capacity of backup infrastructure including DG and UPS is generally provisioned for the peak capacity of the datacenter, since the entire datacenter load is transferred to them upon an outage. It takes about 20-30 seconds for the Diesel Generator to start and generate enough power to source the entire datacenter. In addition to this startup delay, additional delay is incurred when transferring the load from UPS to DG, which is generally performed in gradual load-steps, making the overall transition delay to ˜2-3 mins [8]. This translates in to a requirement of at least 2 minutes UPS battery runtime. It is important to note that even before starting to use the DG, the datacenter would have restored utility power for more than 30% of the power outages (see Figure 1).

UPS units come with certain base energy capacity whose exact quantity depends on the battery technology and the provisioned peak power capacity. For instance, 2-10 KW lead-acid batteries come with a base energy capacity of ˜2-4 mins. This corresponds to where the certain energy storage technology (lead-acid batteries in this case) falls in the Ragone plot (a plot of power versus energy densities as discussed in [63]), wherein a required power from a given technology also determines its energy capacity (and vice-versa). Consequently, while composing the battery cells to achieve a certain amount of battery power, we would automatically get some amount of inherent base battery energy capacity for free. Additional battery modules can be added to this base capacity depending on the energy requirement.

The amount of time taken to drain the battery, i.e., runtime, is not a linear function of the load imposed on it. For illustration, we provide the runtime chart for an APC 4KW battery in Figure 3. Runtime is disproportionately higher at lower load levels. For instance, while the battery shown in the figure can last for 60 mins at 25% load (1000 W) effectively delivering 1kWh of energy, it lasts only for 10 mins at 100% load (4000 W) delivering 0.66kWh of energy. We exploit this crucial property to extend battery runtime during power outages.

Cost Models: The capital expenditure (cap-ex) of the DG and UPSes includes the up-front procurement cost and the amortized future replacement cost. We express cap-ex as amortized $/year, using a linear depreciation model. The operational expenditure (op-ex) includes the cost related to using these devices, corresponding to the cost of diesel fuel and efficiency for the DG and the energy losses for the UPS. The op-ex cost is likely to be negligible since these are rarely called upon, compared to the cap-ex, and we consequently focus on the latter.

The DG cap-ex, *DGCost* (in $/year) is linear with respect to its provisioned peak power capacity, *DGPowerCapacity* (in KW), and can be expressed as:

$$DGCost = DGPowerCost \times DGPowerCapacity \quad (1)$$

where *DGPowerCost* (in $/KW/year) is the amortized cost of diesel generators per unit capacity.

The UPS cap-ex, *UPSCost* (in $/year) depends on (i) provisioned peak power capacity (*UPSPowerCapacity* in KW) and (ii) provisioned battery energy capacity, *UPSEnergyCapacity* (in KWh), both with corresponding per unit costs of *UPSPowerCost* and *UPSEnergyCost* in $/KW/year, respectively. Recall that we get a base energy capacity for free when provisioning the batteries for a given *UPSPowerCapacity*. Hence we subtract that cost, effectively implying that the energy related cost is only incurred for the extra energy capacity required in addition to the base capacity:

$$\begin{aligned} UPScost = {} & UPSPowerCost \times UPSPowerCapacity + \\ & UPSEnergyCost \times (UPSEnergyCapacity \\ & - (UPSPowerCapacity \times FreeRunTime)) \end{aligned} \quad (2)$$

[3] Although UPS units are also used to handle brief periods of brownouts and frequency/power sags and swells, in the context of this paper, we include these events as power outage events since there is no inherent difference in the way UPS is used for handling these events.

where *FreeRunTime* refers to the run time expected at base energy capacity at rated power.

The total backup infrastructure cost is simply the sum of DG and UPS costs. Table 1 lists the values of the parameters in these equations for current technology. The DG and UPS power/energy costs and the *FreeRunTime* values are obtained from [7]. The battery *FreeRunTime* assumes the rack-level battery placement and rated power capacity to supply a rack (in multiple of kilo-watts). We have also evaluated server-level battery configurations and conducted sensitivity analysis on *FreeRunTime*, and the corresponding results can be found in [64]. The cost values of DG power, UPS power and UPS energy are depreciated based on the lifetime of these components: 12 years for DG lifetime and UPS power electronics, and 4 years for lead-acid batteries [11].

Parameter	Value
DGPowerCost	$83.3/KW/year
UPSPowerCost	$50/KW/year
UPSEnergyCost	$50/KWh/year
FreeRunTime	2 min

Table 1. DG and UPS cost estimation parameters [7, 11]. All cost values are depreciated based on a DG lifetime of 12 years and UPS battery lifetime of 4 years.

Peak Power (MW)	DG cost (per year)	UPS runtime (minutes)	UPS cost (per year)	Total cost (per year)
1	0.08 M$	2	0.05 M$	0.13 M$
10	0.83 M$	2	0.51 M$	1.34 M$
10	0.83 M$	42	0.83 M$	1.66 M$

Table 2. Estimated amortized cap-ex annual cost of backup infrastructure for different datacenter capacities. M$ indicates million dollars.

Table 2 shows the backup infrastructure cost at different peak power and UPS energy capacity (expressed as runtime) requirements. Three interesting observations are: (i) backup infrastructure costs amount to several million dollars of capital investment for multi-megawatt datacenters, (ii) while the backup infrastructure cost varies almost linearly with peak power, it rises very slowly with provisioned energy capacity (for instance, a 20 fold increase in UPS energy translated to just 24% increase in overall cost), and (iii) for less than 40 minutes of outage, the cost of using UPS batteries is lower than that of DG. Additionally, battery cost is continuing to drop due to its recent proliferation in commodity products (while DG cost has remained relatively stable), indicating that batteries will become more favorable in future.

4. Cost-Performance-Availability Tradeoffs

We now use the cost model to identify various underprovisioning options for the backup infrastructure and discuss their performance and availability implications. Recall that we use the term *performability* to loosely indicate both performance and availability during power outages.

Configuration	DG Power	UPS Power	UPS Energy	Cost
MaxPerf	1	1	2 mins	1
MinCost	0	0	0 mins	0
NoDG	0	1	2 min	0.38
NoUPS	1	0	0 mins	0.63
DG-SmallPUPS	1	0.5	2min	0.81
SmallDG-SmallPUPS	0.5	0.5	2 mins	0.5
SmallPUPS	0	0.5	2 min	0.19
LargeEUPS	0	1	30 min	0.55
SmallP-LargeEUPS	0	0.5	62 min	0.38

Table 3. Different options for underprovisioning backup infrastructure. Cost is normalized to the current datacenter practice (*MaxPerf*). "P" in "SmallP-" stands for Power; "E" in "LargeE-" represents Energy.

Table 3 presents different backup infrastructure configurations with varying DG and UPS capacities, along with their cost estimates. *MaxPerf* indicates the current datacenter practice with both DG and UPS provisioned with full power capacity (equivalent to datacenter peak requirement) and offers seamless performance during power outages. *MaxPerf* uses UPS batteries merely as a transition mechanism to DG which amounts to ˜2 mins of UPS battery runtime (refer Section 3). Since *MaxPerf* is popular with today's datacenters, we use it as a performability and cost baseline for comparison. The second configuration, *MinCost* in Table 3, points to the other extreme, where the datacenter is completely unavailable (offering no performance) during power outages. *MinCost* does not provision any backup infrastructure, and thereby not incurring any associated costs.

Datacenters may wish to operate in between these two performability-cost extremes. The ideal operating point would have the maximum performability as that of *MaxPerf* and the minimum cost as that of *MinCost*. There are different intermediary backup configurations with varying degrees of underprovisioning, that offer different trade-offs. Table 3 lists some of these, along with consequent cost benefits. Eliminating DG in *NoDG* results in 62% cost reduction compared to current practice (*MaxPerf*). Removing UPS in *NoUPS* translates to 37% savings. While underprovisioning UPS power capacity in *DG-SmallPUPS* saves 19%, underprovisioning both DG and UPS power capacity in *SmallDG-SmallPUPS* costs 50% less. *SmallPUPS* achieves 81% cost savings by eliminating DG and underprovisioning UPS power capacity. *LargeEUPS* shows an interesting configuration where even after increasing the UPS energy capacity to 30 minutes (a factor of 15 increase from MaxPerf), it still saves 45% of the cost by eliminating DG. More interestingly, *SmallP-LargeEUPS* achieves the same cost as *NoDG* (38% of *MaxPerf*) by trading power capacity for longer run time.

Performability during Power Outages Underprovisioned backup infrastructure has multiple performance and availability ramifications: (i) degraded performance due to *re-*

duced power capacity of backup infrastructure, (ii) service or application unavailability (no performance) due to *insufficient backup energy capacity* to last for the duration of the outage, and (iii) impact to both performance and availability due to loss of application state.

Loss of application state, such as volatile application data in CPU registers, caches, and DRAM, can continue to impact application performance even after power is restored (either via utility or DG) . For instance, an outage in datacenters without UPS (e.g., *MinCost* and *NoUPS* in Table 3) will result in immediate loss of volatile application state (server/application crash). This loss continues to impact performance and availability due to several reasons including: (a) re-initialization of various server components such as CPU, disks and re-establishment of network sockets, external service authorizations etc; (b) consistency checks of various software components like file systems, networked systems etc., due to potential loss of partially committed state; (c) re-loading of the entire OS and application stack from disk to memory; (d) application specific warm-ups which includes pre-loading or pre-computing certain application state to improve performance, especially for applications that use significant memory such as Memcached, in-memory index search etc.; and, (e) re-computation of work that was earlier committed to memory but not persisted to disks. The above overheads may either manifest as extended unavailability beyond the power outage duration ((a), (b) and (c)) or as degraded performance upon resuming application execution after the power outage ((d) and (e)).

To summarize, performability of an application during power outages depends not just on the ability to sustain application execution but also on the ability to preserve application state. In the next section, we discuss the cost implications of various system techniques that allow us to either sustain application execution or preserve application state during outages.

5. Handling Outages With an Underprovisioned Backup Infrastructure

Given the above discussion, we classify system techniques to handle outages into two broad categories, *sustain-execution* and *save-state* as shown in Figure 4, and discuss them below.

Technique	Time to take effect	Power after activation
Throttling	Tens of μsecs	Throttled state
Migration	Few mins	Consolidated state
Proactive Migration	100ms-few secs	Consolidated state
Sleep	˜10 secs	2-4W per DIMM
Hibernation	Few mins	0 Watts
Proactive Hibernation.	Few mins	0 Watts

Table 5. Impact of system techniques on backup infrastructure capacity.

Sustain-execution: These techniques allow us to continue executing the application even after a power failure, possibly at a lower power draw to stay within the underprovisioned backup infrastructure capacity (power or energy).

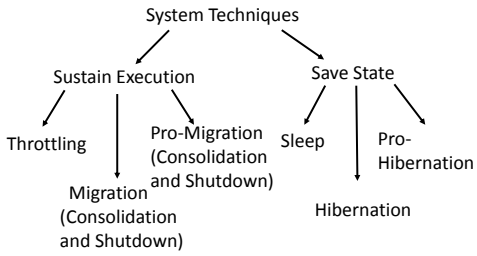


Figure 4. Techniques for realizing different application performability goals during power outages.

Throttling: The application may be operated in a lower performance mode using active CPU power states (P/T states) [20, 24, 36, 54, 67]. Transitioning to these throttling states is almost instantaneous [4] (within tens of μsecs) and therefore acts as an effective technique to reduce the peak power requirement from the backup infrastructure.

Migration (Consolidation and Shutdown): Throttling can reduce power but incurs the idle power penalty of keeping all servers active. An alternative is to consolidate applications on fewer servers, with each given a smaller fraction of the processor cycles, memory space and other resources [16, 26, 35, 52, 59, 69]. This can be more energy proportional for current technology. In this approach, immediately after a power failure, application state is migrated to a remote server while keeping the original server powered until migration is complete. Once all state is transferred, the originating server is powered down and the application is resumed at the remote server. In this approach, we assume the persistent state of the application is available in a shared storage server which continues to have power backup even when the power backup is under-provisioned. Hence, only the volatile state needs to be migrated. We use the existing implementation of live-migration in Xen for evaluating this technique [18].

Proactive Migration (Consolidation and Shutdown): The time required for migration depends primarily on the size of volatile application state. Datacenter applications may have huge in-memory application state (as high as the memory capacity - between ˜64 to 128GB for today's datacenter servers) which can result in large migration times (potentially exceeding the outage duration). This technique attempts to reduce the amount of state that needs to be moved after a power failure by periodically flushing application memory state to remote server memory [56] during normal operation (when utility power is active). Successive copying only needs to move the memory state that has been modified since the last migration (similar to how live migration

[4] Recall from section 3 that servers have ˜30ms of power supply capacitance which acts as a ride-through before transfer to backup infrastructure. This duration can be used to transition the server to the throttled state.

Technique	Normal operation	Start of utility outage	During utility outage	After utility restored
Maxperf	Full service	Full service	Full service	Full service
Mincost	Full service	Server/App crash	No service	Server/App Restart
Throttling	Full service	Throttled Perf.	Throttled Perf.	Restore full service
Migration	Full service	Migrate to remote memory	Consolidated service	Migrate back
Proactive Migration	Periodic dirty-state flush to remote memory	Migrate remaining dirty state to remote memory	Consolidated service	Migrate back to full service
Sleep	Full service	Suspend to local memory	No service	Resume from memory
Hibernation	Full service	Persist to local storage	No service	Resume from disk
Proactive Hibernation	Periodic dirty-state flush to local storage	Persist remaining dirty state to local storage	No service	Resume from disk

Table 4. Performance and Availability implications of underprovisioning techniques.

works). This reduces the amount of (volatile) state that needs to be transferred after a power failure and hence reduces the amount of backup energy capacity required for the migration. We leverage existing implementation of periodic virtual machine checkpointing in Remus [19] for evaluating this technique.

Save-state Immediately after a power failure event, the application state is preserved either by pushing the state to memory (while ensuring DRAM is supplied sufficient power to retain data) or to local persistent storage. The servers themselves stop continuing to execute application code.

Sleep: The application and the OS stack is suspended and the server enters the S3 (suspend to RAM) state where the volatile memory (DRAM) is operated in self-refresh mode and all other server components are turned off [1, 41–43, 55, 68]. Though this technique does not offer any application service during the power outage, the resume time is fast after power is restored (only the processor caches need to be loaded).

Hibernation: The application state is pushed to local persistent storage. Unlike *Sleep*, this allows completely powering down the servers once the state is pushed to the disk, but comes with a higher restore latency.

Proactive Hibernation: The modified volatile state of the application is periodically pushed to local persistent storage during normal operation (active utility). This may reduce the amount of state that needs to be pushed after a power failure compared to Hibernation and therefore may require less backup capacity for persisting the state.

Nearly all the hardware capabilities and APIs needed to implement these techniques are available in today's systems, though some may need to be tuned for our usage scenario, e.g., repeated copying of dirty state in live-migration can be tuned to reduce the overall migration time and hence the backup energy; the Remus technique takes a consistent snapshot by suspending the entire application whereas we are only interested in minimizing the amount of dirty state to be transferred after a power failure and do not care about the remote state being consistent at other times. In this paper, we use the available capabilities as-is and therefore present a conservative estimate of the efficacy of the techniques.

It is easier to visualize the contrasting performability implications of these system techniques during a power failure using four operational phases as identified in Table 4: (a) Normal operation – utility is active and powering the datacenter, (b) Start of Power Failure – activities performed immediately at the start of a power outage, (c) During Power Failure – activities performed during the power outage, and, (d) Power Restored – activity performed after power is restored. Along with the techniques discussed above, Table 4 also presents the performability offered by today's approach, *MaxPerf* and the zero cost baseline, *MinCost*. While the *sustain-execution* techniques continue to offer performance even during power failure, they may not have enough backup capacity to sustain the energy needs throughout the entire outage duration. On the other hand, the *save-state* techniques significantly reduce the energy capacity requirement from the backup infrastructure, but they do not offer any performance during power failure.

We also summarize the demand imposed by these techniques on the backup infrastructure capacity in Table 5. The amount of time required for the technique to take effect after a power failure, together with the power requirement after the technique is enforced, help quantify the required backup infrastructure energy and power capacity.

Hybrid Techniques The ability of the *sustain-execution* techniques to continue serving applications at low power during outages and the ability of *save-state* techniques to preserve application state at almost no power cost, suggest the possibility of combining them for better cost-performability tradeoffs.

The peak power capacity of the backup infrastructure is a crucial factor in determining the overall cost (for both UPS and DG as shown in Table 2). Among the basic techniques identified above, *Throttling* is the only technique that is guaranteed to reduce the peak power (refer Table 5) (even migration for subsequent shutdown can create a momentary spike). The other techniques are primarily geared towards reducing the energy requirement from the backup infrastructure and can be combined with *Throttling* to reduce the backup infrastructure cost. We use the '-L' notation (denoting low power) along with the name of the basic technique in Table 6 to refer to these hybrid combinations. The con-

Hybrid technique	During power failure
Sleep-L	Throttle while going to sleep
Hibernate-L	Throttle while going to hibernate
Throttle+Sleep-L	Throttle + throttle while going to sleep
Throttle+Hibernate	Throttle + throttle while going to hibernate
Migration+Sleep-L	Migrate + throttle while going to sleep

Table 6. Hybrid Sustain-Execution + Save-State techniques.

sequent reduction in peak power could possibly come at the cost of higher energy required from the UPS, but still reduce overall backup cost. For instance, hibernation may take a long time to complete when combined with throttling but the overall UPS cost may still be lower due to the asymmetry between UPS power and energy costs. We investigate these trade-offs and the efficacy of the hybrid combinations enumerated in Table 6 in our evaluations.

6. Experimental Evaluation

Experimental Setup and Methodology: We use identical dual socket servers with 6-core 3.4 GHz Intel processors (12 cores per server), 64 GB DRAM, and a 1 Gbps Ethernet interface and run our applications hosted on the Linux OS. The power consumption of each server is monitored using an external Yokogawa high-resolution power meter. The server idle power is around 80W and the peak power draw that we have measured is 250W. The dynamic power consumption can be modulated using 7 voltage/frequency P-states and 8 clock throttling T-states.

One would ideally like to carry out our experiments on a datacenter scale platform with all the backup infrastructure in place. However, a smaller setup can be used to glean nearly all the insights as that from a large scale platform, without explicit backup equipment such as UPSes or DGs, using the following methodology. We subject each application or server to the different system techniques described earlier and record the power consumption (both peak power and energy) using the external power meter. Along with power data, we also collect the application performance and down time, both when the application is subjected to the system techniques (during the assumed outage duration) and and when the application resumes normal operation (immediately after the assumed outage duration). The outage start and end times are noted. Power data collected at fine temporal resolution allow us to calculate the required DG and UPS power and energy capacities for each evaluation run.

Implementation of System Techniques: We build on existing functionality in current systems to implement these techniques as below. *Sleep* and *Hibernation*: standard OS commands in Linux; *Throttling: cpufreq* driver in Linux; *Migration (Consolidation):* We run applications on a Linux Virtual Machine (VM) with 28 GB of allocated physical memory. We leverage Xen live migration [18] and Remus implementations [19] for our migration related experiments. We use a relatively aggressive consolidation by powering down every alternative server, reducing the number of servers to half the original size; *Proactive Hibernation* and *Proactive Migration*: for each application, we profile the frequency at which its memory pages are being modified (analogous to the dirty bitmap used in live migration [18]) and estimate the amount of memory pages that need to be written to disk (in case of proactive hibernation) or copied to remote memory (in case of proactive migration). We limit the frequency of the periodic modified-page copying operation in order to avoid any perceivable performance impact during normal operation.

Workloads: We consider the following workloads (Table 7) that have different kinds of state, and consequently demands on the backup infrastructure for availability:

- Specjbb [58] emulates a supermarket retailer IT infrastructure using a three-tier architecture comprising web, application, and database tiers. We execute the benchmark and all its tiers on a single server. Specjbb uses an in-memory database, which has both read-only and modified data. Losing volatile state during a power failure may cause Specjbb to recompute lost computation and impacts its throughput.
- Web-search is an internally developed workload that emulates the index searching component of search engines. Web-search stores several hundred gigabytes of index data in persistent storage and uses volatile memory (DRAM) as a cache for frequently accessed index data – around ˜40 GB index data in memory for our experiments. We use a real world query trace to generate client traffic. This workload measures performance as aggregate throughput (queries per second) that can be achieved by the server within a high-percentile latency constraint. The index data which occupies the major portion of server memory is read-only and can be read-back from persistent storage if volatile state is lost during a power failure with a consequent performance penalty.
- Memcached [44] is an in-memory key value store for small chunks of data. It primarily uses volatile memory to improve read performance. We use a read-only client workload to exercise the data and use throughput as its performance metric.
- SpecCPU benchmarks [57] represent High Performance Computing (HPC) applications. These applications may run for hours or even days. If volatile state is lost due to power outage, these applications will typically have to recompute the lost state (one can alleviate the performance impact by checkpointing partial results). We use *mcf* from SpecCPU2006 in our experiments as a representative for memory intensive scientific computation workloads. Since each *mcf* instance only consumes ˜2GB memory, we instantiate multiple *mcf* instances to increase its memory usage to emulate large memory footprint HPC applications.

Workload	Memory Usage	Performance Metric
Web-search	40 GB	Latency-constrained, queries/sec
Specjbb	18 GB	Latency-constrained, ops/sec
Memcached	20GB	Queries/second
SpecCPU (mcf*8)	16GB	Completion time

Table 7. Workloads Description

Evaluation Metrics: We consider backup infrastructure cost, application performance and availability as metrics for evaluating the efficacy of our underprovisioning techniques.

- *Cost:* We use backup infrastructure (UPS and DG) capex based on the cost model in Section 3. We normalize the cost of the different under-provisioned configurations to that of today's datacenter configuration (*MaxPerf* in Table 3).
- *Down time:* We report the total time for which an application is unavailable (not performing computation or responding to users) during a power outage and immediately after power is restored.
- *Performance during power outage:* Table 7 lists the application specific performance metrics. For each application, we normalize its performance to that in *MaxPerf*. Since performance may continue to be impacted after power restoration for different lengths of time under different system techniques, we report performance impact over a common duration, the power outage duration. We report the impact beyond the outage as *down time* and distinguish between actual down time and performance induced down time.

6.1 Tradeoffs Between Backup Configurations

We first evaluate the cost and performability tradeoffs for different power backup configurations (i.e., different DG power capacities and UPS power and energy capacities) from Table 3. For each backup configuration, we choose the system technique (from Tables 4 and 6) that offers the highest performance and lowest down time.

Figure 5 presents the cost, performance and down time for *Specjbb* for various backup configurations. While today's approach, *MaxPerf*, offers the best performance and zero down time for all outage duration, the *MinCost* configuration offers no performance during an outage and suffers significant down time – as much as 400 seconds even for a short 30 seconds outage. The high down time suffered by *MinCost* is due to the server restart time, creation of Specjbb processes and time to catch up to the target throughput due to the loss of state.

The other configurations cover a large spectrum of offered performance and availability:

Impact of DG: For configurations with DG (*NoUPS* and *DG-SmallPUPS* in Table 3), long outages can be transformed into short outages from the perspective of performability since DG will supply power after the initial 2-minute start-up delay. In *NoUPS* (not shown for clarity), the down-time is same as that for *MinCost* in Figure 5(c). *DG-SmallPUPS* can ride-out the DG start-up delay with zero downtime but with a performance penalty since the smaller UPS implies reduced performance during DG start-up. Performance should not be compared across multiple outage duration since each one is normalized to the baseline for that duration, and different outage handling system techniques could have been selected depending on outage duration.

The smaller UPS capacity required is only 15s of runtime at 50% of *MaxPerf* power (using *sleep-L* as the outage handling technique for the DG start-up delay), but the minimum battery capacity dictated by the Ragone plot leads to higher battery runtime. The cost reduction of *DG-SmallPUPS* is hence only 20% compared to *MaxPerf*.

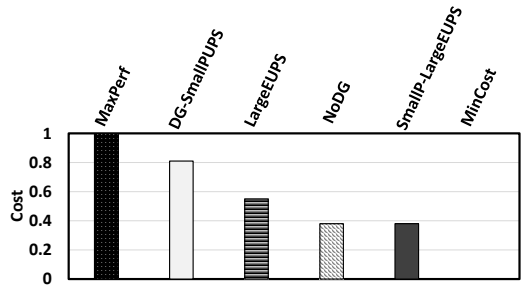
SmallDG-SmallPUPS (not shown for clarity) in Table 3 also has zero down time but at higher performance penalty. Configurations without DG cannot avoid down time during long outages.

Impact of UPS: UPS acts as a low-cost alternative to DG for handling short outages, that comprise the majority of outages. Consider configurations that completely eliminates the DG: *NoDG*, *LargeEUPS*, *SmallP-LargeEUPS* in Figure 5. While the *NoDG* configuration with limited UPS capacity suffers degradation in both performance (drops to 60% of *MaxPerf*) even for a 5 mins outage, one can significantly improve the performability by purchasing more UPS energy capacity. For instance, *LargeEUPS* with 30 minutes of UPS battery capacity achieves the same performance as *MaxPerf* upto 30 mins outage duration and sustains 60% of (degraded) performance for upto 1 hour outage duration, at only 55% of the cost of *MaxPerf*. Although *NoDG* and *SmallP-LargeEUPS* incur the same cost (38% of *MaxPerf*), the latter achieves better performability than *NoDG* (which do not sustain beyond 5 mins outage) for 30 mins or longer outages, by trading peak power (reduced by half) for higher energy (runtime of 62 mins).

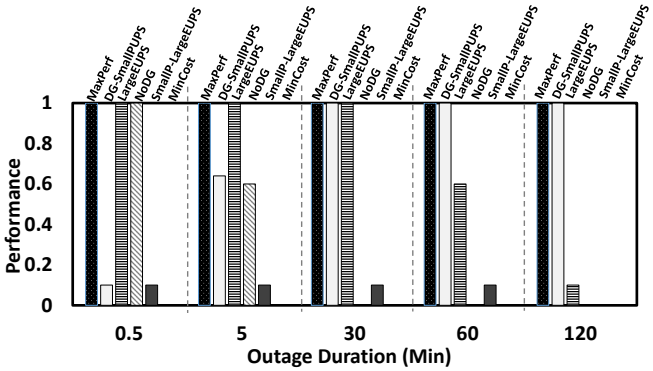
Summary of Insights: (i) Though DG translates long outages into small ones from the perspective of offered performability, it does so at a significant cost. (ii) UPS plays a crucial role in improving performability for short outages irrespective of the presence of DG. (iii) UPS can eliminate DG for up to 100 mins of outage duration and offer the same performance as with today's approach at the same cost. (iv) UPS can result in 40% cost savings for outages as long as 1 hour for datacenter willing to tolerate 40% performance degradation during outages. (v) For the same cost, the performability offered by UPS with small power capacity and longer runtime may be better than that offered by UPS with high power capacity and shorter runtime for relatively long outages.

6.2 Effectiveness of Outage Handling Techniques

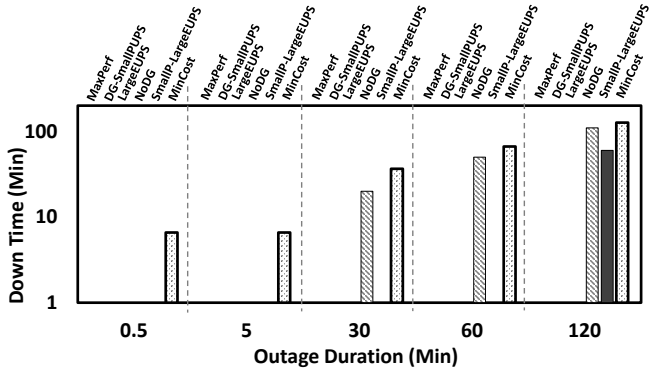
We now compare the performability-cost tradeoffs offered by different system techniques listed in Tables 4 and 6. In



(a) Cost of configurations from Table 3.



(b) Performance of configurations under different outage durations.



(c) Down time of configurations under different outage durations.

Figure 5. Cost and performability tradeoffs between the 6 configurations - *MaxPerf*, *DG-SmallPUPS*, *LargeEUPS*, *NoDG*, *SmallP-LargeEUPS* and *MinCost* - for Specjbb. In (a), we repeat the costs for these configurations from Table 3 for convenience. For each outage duration, in (b) and (c), we give the performance and downtime for these configurations. It is only for outages longer than 60 minutes that the *LargeEUPS* configurations become less attractive.

Technique	Save time	Resume time	Peak power
Sleep	6 secs	8 secs	1
Hibernate	230 secs	157 secs	1
Proactive Hibernate	179 secs	157 secs	1
Sleep-L	8 secs	8 secs	0.5
Hibernate-L	385 secs	175 secs	0.5

Table 8. Time to save and resume *Specjbb* memory state for different techniques. The save power draw (normalized to server peak) of these techniques is also presented.

the interest of clarity, we first show representative results for Specjbb, and then show specific results for the other applications. Recall from Section 6.1 that the presence of DG in the backup infrastructure is not only expensive but is also uninteresting in its performability implications for outages longer than the DG start-up time. Therefore, for the rest of the evaluation, we only consider backup configurations without DG and consider variations in the UPS peak power and energy capacity. For each system technique, we use the lowest cost backup configuration (combination of UPS peak and energy capacity) at each of the offered performance and availability operating points.

Impact of Power Outage Duration: Figure 6 shows the cost of backup infrastructure, application down time and application performance impact of the different system techniques for *Specjbb* application under different outage duration – from 30 seconds to 2 hours. In the case of techniques which employ DVFS throttling, we use two bars in the figure to show the minimum and maximum values (the range) offered based on the chosen voltage and frequency states.

Sustain-execution Techniques: These techniques offer high performance at low cost for short and medium outages, but incur high cost for long outages. For instance, *Throttling* can achieve the same performance as *MaxPerf* at less than 40% of its cost for outage duration up to 30 minutes. For long outages ($>$ 1 hour), realizing similar cost reduction (60% of *MaxPerf*) results in degraded performance for *Throttling* (drops to 60% of *MaxPerf*) and even becomes infeasible to sustain the application beyond 4 hours outage duration. *Migration* techniques are better than throttling for medium to long outage duration since after migration the applications enjoy better performance under the same cost budget, due to lack of energy proportionality in today's servers [10]. However for short and even medium outages, throttling is preferred because migration overheads are high (*Specjbb* takes 10 minutes to migrate) and performance loss due to additional throttling required to suppress power spikes during migration [29]. *Proactive Migration (PM)* can reduce the application state that needs to be migrated after a power failure. For *Specjbb*, this resulted in a reduction from 18GB to 10GB, corresponding to a lower migration time of 5 minutes. This reduction in migration time translates into some cost savings (not visible in the figure).

Save-state techniques: These techniques do not offer any performance during the outage and are better suited to handle short outages where preserving state reduces the additional unavailability after power restoration. For instance, *Sleep-L*, which costs only 20% of *MaxPerf*, has a much lower down time of 38 seconds compared to *MinCost* which has over 400 seconds of down time for the 30 second power outage duration.

Save-state techniques may also incur a penalty to resume application state but this is generally much lower than that of *MinCost*. Table 8 shows the measured save and resume du-

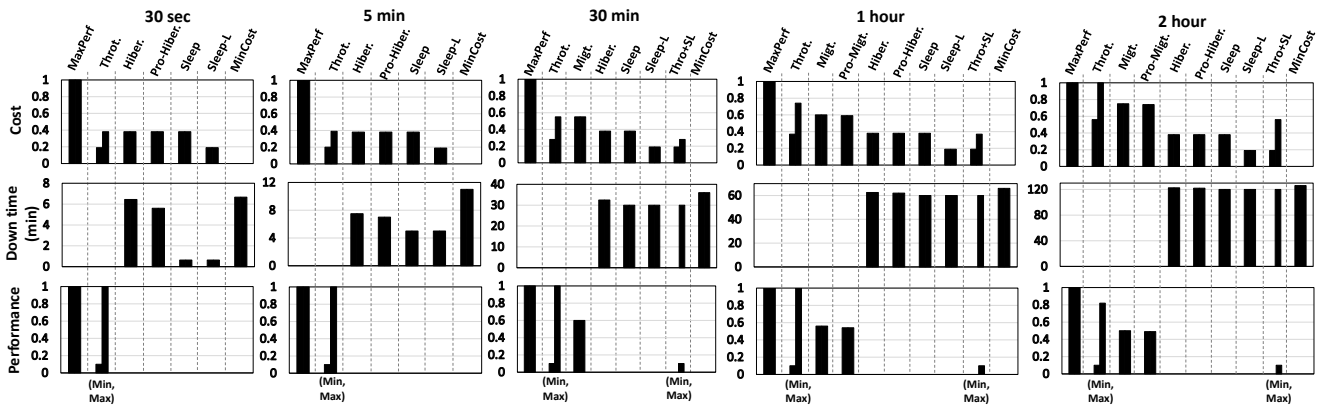


Figure 6. Power outage duration impact on different techniques for Specjbb. In techniques which use DVFS Throttling, the minimum and maximum values are shown to illustrate the range based on the chosen voltage-frequency states.

ration for the *Specjbb* application, along with its peak power draw when operating under these techniques. The low-power (*-L) techniques take a longer time to save the application state, especially for *Hibernate-L* (while *Sleep-L* save time remains unaffected) but reduces the peak power draw by half. While the resume time always manifests as additional down time for the application after power is restored, the save time impacts availability only when the outage duration is smaller than the save time. For instance, *Hibernation* may be a bad idea for a 30 second outage for *Specjbb* as shown in Figure 6. *Proactive Hibernate* can reduce the time required to save state (by 22%), but is still not well suited for short outages.

We find that the most effective technique is different for different outage duration. For short outages (0.5 min to 5 min), *Throttling* can achieve best performability and cost tradeoffs compared to other techniques. For medium outage duration (30 min to 1 hour), *Throttle+Sleep-L* is able to sustain and preserve state for the entire outage duration even with very limited battery capacity. This is primarily due to the extremely low power consumption of *sleep* technique which is around 5W per server and UPS runtimes stretch significant for lower load-levels (refer Section 3). And finally, for long outages (2 hours and beyond), *Throttling* and *Migration* become infeasible for cost less than 56% of *MaxPerf*, since they quickly drain the limited UPS battery capacity and are not able to sustain operation for the entire outage duration. On the other hand, *Throttle+Sleep-L* can sustain at as low as 20% cost. However, for long outages, preserving state may not buy much since the outage duration far outweighs the overheads of losing state. For handling such long outages, request or load redirection to geo-replicated datacenters would be a better solution [3, 32].

Impact of Application Memory Usage: The memory used by an appication impacts the time that it takes to save state, thereby also having a consequence on availability (whether the state can be saved before power runs out) as well as on performance (may need to employ more aggressive performance impacting power saving techniques). We have evaluated such performability-cost tradeoffs for varying memory state size of the *Specjbb* application, over different outage durations. We summarize the results here and the reader is referred to [64] for further details. As the state size reduces, the down time due to *Hibernation* and *Proactive Hibernation* techniques reduces, though the corresponding impact on cost is not perceivable (due to lower battery energy cost). Sleep based techniques (*Sleep* and *Sleep-L*) remain unaffected with application state size. Sustain-execution techniques achieve better performability and lower cost with smaller state size. For *Throttling*, this can be attributed to the lower injected load on *Specjbb* at lower memory footprint size, resulting in less power and energy draw from the UPS. For migration based techniques, smaller state size directly translates to shorter migration time. These insights could be exploited to develop an adaptive online strategy that tracks memory usage and dynamically employs the appropriately effective technique.

Influence of Application Characteristics: We now compare and contrast the effect of underprovisioning for applications with diverse performance and availability (state recovery time) characteristics. We mainly highlight the differences.

Memcached (Figure 7): Since Memcached loads the memory with the necessary data from disk before handling client requests, losing memory state will result in the reloading of the lost data. While the down time is 480 seconds for a 30 seconds outage with the *MinCost* configuration, somewhat surprisingly, the down time is even longer, 1140 seconds, for *Hibernation*. This implies that losing application state after a power failure (and re-loading the data after restoring power) may be more efficient compared to hibernation overheads for applications that directly uses data fetched from disk without any modification. We find that the performance offered by *Throttling* and *Migration* is much better than that for *Specjbb*. We suspect this is attributable to high memory-related CPU stalls for *Memcached* (due to

its random memory access as opposed to *Specjbb*) which is better suited for throttling based techniques [67]. *Proactive Migration* combined with throttling achieves 20% more cost savings compared to *Migration* since proactive migration is able to significantly reduce the application state that needs to be migrated after the power outage. This implies that applications with lower frequency of page modifications may benefit more from the *Proactive Migration* technique.

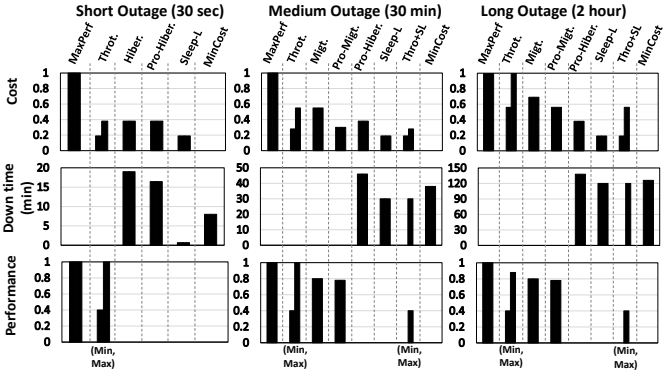


Figure 7. Tradeoffs for Memcached

Web-search (Figure 8): Web-search uses volatile memory as a cache for index data that are stored in persistent storage to improve query latency. Since the index data is read-only in memory, one might think its behavior will be similar to that of *Memcached* where we may afford to lose memory state (and re-load after restoring power) as opposed to persisting state to disk using hibernation. Instead, we find that losing memory state for *Web-search* can be extremely harmful to its performance and availability, especially for short outages as indicated using *MinCost* in Figure 8. For instance, *MinCost* results in a total application down time of 600 seconds as opposed to *Hibernation* which results only in 400 seconds down time for a 30-second outage. The 600 seconds down time for *MinCost* for this workload can be attributed to multiple factors including (i) server restart time ˜2 mins, (ii) index data pre-population ˜3.5 minutes and (iii) application warm-up duration ˜4-5 minutes. Though the web-search workload is available after 5.5 minutes once power is restored, the queries suffer poor performance (almost 30-50% reduction in throughput) during the first 4-5 minutes (warmup duration) which we report as additional down time. Similar to other workloads, sleep when combined with throttling is an effective technique to achieve a good cost-performability tradeoff.

SpecCPU (Figure 9): These scientific applications may run for several hours before computing the end result and any loss of power during the execution may result in re-computation. We consider the amount of time required for re-computation after power is restored as part of down time and report performance only during the power outage duration. Depending on when the outage occurs during the application execution, the impact on down time can span a large

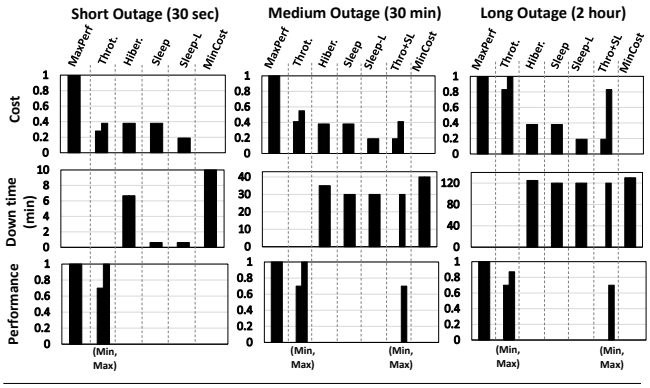


Figure 8. Tradeoffs for Web-search

range for *MinCost* as shown in Figure 9. We find the tradeoffs between other techniques very similar to that of Specjbb application.

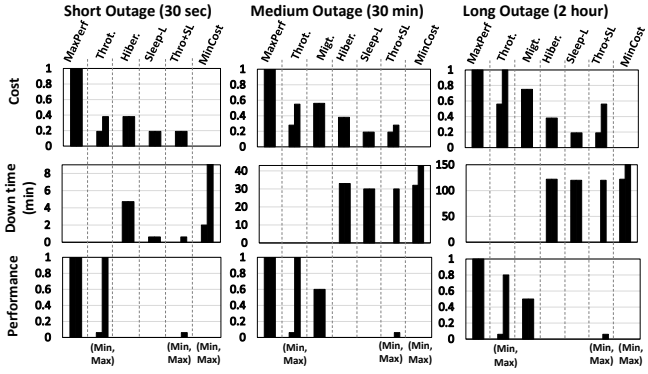


Figure 9. Tradeoffs for SpecCPU (mcf*8)

Summary of Insights: (i) Sleep is a low cost technique for achieving lower application down time for short to medium outages. (ii) Throttling can cover a large spectrum of cost-performability for short to medium outages, though it becomes infeasible at lower cost budgets. (iii) Migration/consolidation is preferred for longer outages due to better performability compared to throttling (owing to lack of energy proportionality in today's servers). (iv) Hybrid techniques allows us to traverse the entire cost-performability spectrum even for long outages. (v) For very long outages ($>$ 4 hours), it is preferred to transfer load (request redirection) to geo-replicated datacenters if no DG is used. (vi) Application state size crucially impacts the performability-cost tradeoffs associated with techniques such as *Hibernation* and *Migration*.

7. Discussion and Future Enhancements

TCO Considerations: In addition to the power infrastructure related costs, an organization may need to base its decisions on a more holistic picture, including any revenue loss as a result of unavailability. Though such matters are very organization-centric, we illustrate how such an anal-

ysis could be done for an organization such as Google, where a large majority of its revenue comes from its datacenter operations. It has been reported that the power capacity across Google Datacenters was around 260 MW [31] in 2011, and Google's revenue in 2011 was about $38 billion [25]. Conservatively, assuming all of its revenue as coming from datacenter operations, we can calculate its revenue/kilowatt/minute as $0.28/KW/min. In addition to this revenue loss during an outage, the corresponding servers are also idle, and hence its capital expenditures (assumed cost of $2000 that is depreciated over a 4 year lifetime) should also be counted as a loss during the outage. This gives a number of $0.003/KW/min. Adding the revenue and capital losses, we can then capture the loss as a function of the minutes of unavailability as is shown in Figure 10. In the same Figure, we also plot the cost savings by not provisioning Diesel Generators (amortized over a lifetime of 12 years) as a horizontal line. All values to the left of the cross-over point (which turns out to be around 5 hours per year in this case) indicate profitable operations despite the corresponding revenue loss. An organization can use this to figure out whether to not provision DGs, or whether to provision additional UPS capacities while not provisioning DGs, or whether to provision full backup capacity. Such decisions would also depend on the organization's ability to seamlessly migrate operations to a different geo-separated datacenter (as discussed next). As mentioned, this is only to illustrate how different organizations may be able to make decisions based on the trade-offs associated with backup under-provisioning.

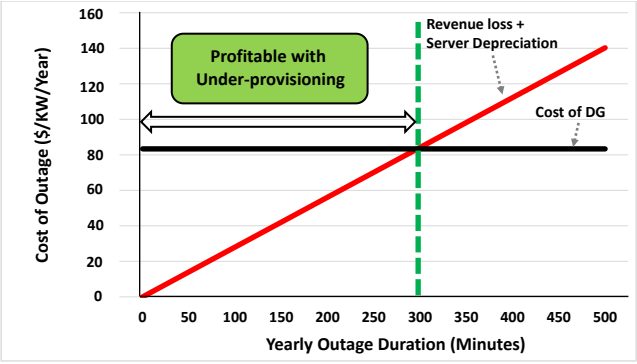


Figure 10. Revenue loss and server depreciation vs. savings from backup under-provisioning for Google 2011 data.

Suitability for Different Datacenters and Organizations: The results of such cost-performability analysis of backup infrastructure are relevant to a spectrum of datacenter sizes and types, since backup costs can be considerable across the spectrum. Further, many organizations today are inherently building multiple geo-distributed datacenters to handle different failures (not just power outages, but network failures, natural disasters, etc.). When geo-distribution/replication is inherent, our work suggests we could leverage these existing multi-datacenter operation capabilities to under-provision or even remove backup capacities in one or all these sites rather than build a highly outage-resilient single datacenter. At the other extreme, one could argue the need for any backup at all (whether DGs or UPS) with Geo-replication. In such cases, power outages can cause load increase at failed-over site, unless adequate spare capacity is set aside. Our solution of having some backup (UPS) capacity, can instead handle a bulk of these outages without requiring a fail-over, since a majority are of short duration. However, not all organizations may have geo-replicated datacenters. Such organizations could leverage cloud services from an external provider upon an outage (and when local backup runs out of energy capacity), to achieve the same functionality. Investigating the mechanisms, performance and costs for this, together with achieving this fail-over within the limited backup capacity, is part of future work.

Challenges: We pose two main challenges in translating the insights gained from our evaluation to an online solution and discuss possible directions.

How do we deal with unknown outage duration? We observed that the preferred system technique for an application with an under-provisioned backup depends on the duration of the outage. When a utility supply fails, it may not be possible (except for planned outages) to know the duration of the outage a priori. One option is for datacenters to use the historic utility outage data from their utility to construct an online predictor (e.g., an online Markov chain based transition matrix of different duration), and use the evolving outage to make dynamic decisions. For instance, with a outage distribution as in Figure 1, we may choose to start with the *throttling* at full performance mode (assuming outage will be short) and gradually transition to lower power modes and then finally (when outage exceeds 5 mins) use the *sleep* or *hibernate* techniques which are known to considerably reduce backup energy requirement.

How do we provision for heterogeneous applications? We found that application characteristics - state size, time for persisting state and restoring execution including subsequent warm-up, performance impact due to throttling, etc. - impact the performability, and hence the choice of the mechanisms and configurations when under-provisioning the backup. Given the diversity of such applications hosted in datacenters, understanding these characteristics, provisioning the infrastructure appropriately (which is done when datacenters are built), and adapting the techniques for these applications can pose interesting challenges. Multiple datacenters or sections in a datacenter could have different backup configurations, in the spectrum of cost-performability choices we outlined. Capacity planning could depend on historic data about multiple application requirements and cost preferences. With heterogeneity in the backup capacity, assigning and migrating workloads to the right infrastructure will help handle multiple types of outages.

Promising Enhancements: We discuss three technology advancements that can enhance the cost-performability tradeoffs for handling power outages.

NVDIMM: Recently, NVDIMMs [48] have been proposed as an alternative approach to persisting application state upon a power outage without the need for UPS. These NVDIMMs have a super-capacitor backed up DRAM+NAND Flash in the DIMM socket and a FPGA controller that transfers data from DRAM to Flash after a power outage without the need for any external backup power source. Though the NVDIMM does not offer any service during a power outage, it can be combined with other backup infrastructure options (similar to the options presented in Table 3). One crucial aspect of NVDIMMs is that the energy storage is localized to the volatile memory (and not the entire server) and it allows procrastinating the save/sustain operations – unlike today's UPS-based approach where the entire server load (even when throttled) is transferred to the UPS immediately after the power failure. This allows significant scope for underprovisioning the backup infrastructure since the load can be gradually shifted to the backup infrastructure (UPS unit) from these already persisted NVDIMM servers.

RDMA over Sleep: As high speed networks (dark fiber, infiniband, etc) become increasingly commonplace both within and across datacenters, RDMA techniques become more and more appealing for faster remote memory access. This will have a crucial bearing on how application state is transferred across machines (both within and across datacenters) during a power outage. For instance, the low cost *sleep* technique used in this paper does not offer any performance. But it can be combined with RDMA capability to access the memory state (on demand) from a remote server while keeping the server processors shutdown with only the memory controller active, similar to the recently proposed *barely-alive* memory servers [6]. NVDIMM can be combined with RDMA to effectively handle very long power outages.

Newer Battery technologies: Li-ion battery is gaining popularity due to its longer lifetime and higher power density. It offers different peak-power vs energy tradeoffs [63], compared to lead-acid (energy is more expensive for Li-ion than power). This may impact the cost-performability tradeoffs offered by the system techniques. For instance, the higher energy cost may prefer more energy saving techniques such as *proactive hibernation* and *proactive migration* compared to peak reduction techniques such as *Throttling*.

Evaluating these ideas opens interesting avenues for future work, and the insights from this paper would help drive such efforts.

8. Concluding Remarks

Towards realizing a low cost backup infrastructure, we have identified different underprovisioning configurations by varying the power and energy capacity of the Diesel Generator (DG) and UPSes, together with system techniques that offer an entire spectrum of cost, performance and availability operating points. We observed several interesting insights based on our evaluation using diverse datacenter applications, subjected to a variety of power outage duration. We found that the UPS plays a crucial role in ensuring high performance and availability for short (few seconds) to medium outages (tens of minutes), irrespective of the presence of DG. We also found that UPS provisioned with higher energy capacity (runtime) can realize significant cost savings by eliminating DG, while offering the same level of performance for outage duration up to 40 minutes. Further, we have shown that applications willing to tolerate performance impact during power outages can combine power reducing techniques such as workload throttling with state-preserving techniques like sleep or hibernation to sustain outages as long as 2 hours at a cost much lower than that of today's approach. We believe the insights gained from this paper can greatly influence the design of future under-provisioned backup infrastructure and development of techniques that allow applications to seamlessly operate during the power outages.

Acknowledgments

We would like to thank our shepherd, Dr. Luiz Barroso, for his generous help and valuable inputs. We also thank the anonymous reviewers for their insightful comments and suggestions. This work has been supported in part by NSF grants 1302225, 1302557, 1205618, and 1213052.

References

[1] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, and R. Gupta. Somniloquy: augmenting network interfaces to reduce pc energy usage. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation (NSDI)*, 2009.

[2] AGIGARAM: DDR3 Non-Volatile DIMM. `http://www.agigatech.com/pdf/pdf_ProductBrief_DDR3_12-0820.pdf`.

[3] M. K. Aguilera. Tutorial on geo-replication in data center applications. In *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems (SIGMETRICS)*, 2013.

[4] B. Aksanli, J. Venkatesh, L. Zhang, and T. Rosing. Utilizing green energy prediction to schedule mixed batch and service jobs in data centers. In *Proceedings of the 4th Workshop on Power-Aware Computing and Systems (HotPower)*, 2011.

[5] H. Amur, R. Nathuji, M. Ghosh, K. Schwan, and H.-H. S. Lee. Idlepower: Application-Aware Management of Processor Idle States. In *Proceedings of the Workshop on Managed Many-Core Systems (MMCS, in conjunction with HPDC)*, 2008.

[6] V. Anagnostopoulou, S. Biswas, H. Saadeldeen, A. Savage, R. Bianchini, T. Yang, D. Franklin, and F. T. Chong. Barely alive memory servers: Keeping data active in a low-power

state. *Journal on Emerging Technologies in Computing Systems*, 8(4), 2012.

[7] APC InfraStruxure Total Cost of Ownership, 2013. `http://www.apc.com/tools/isx/tco/index.cfm`.

[8] APC White paper: Comparing UPS System Design Configurations, 2008.

[9] A. Bar-Noy, M. P. Johnson, and O. Liu. Peak Shaving Through Resource Buffering. In *Workshop On Approximation and Online Algorithms (WAOA)*, 2008.

[10] L. A. Barroso and U. Holzle. The Case for Energy-Proportional Computing. *Computer*, 40(12), 2007.

[11] L. A. Barroso and U. Holzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 2009.

[12] Benefits OF Using Mobile Transformers and Mobile Substations for Rapidly Restoring Electrical Service, 2006. `http://energy.gov/sites/prod/files/oeprod/DocumentsandMedia/MTS_Report_to_Congress_FINAL_73106.pdf`.

[13] O. Bilgir, M. Martonosi, and Q. Wu. Exploring the Potential of CMP Core Count Management on Data Center Energy Savings. In *Workshop on Energy Efficient Design*, 2011.

[14] P. Bohrer, D. Cohn, E. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, R. Rajamony, F. Rawson, and E. V. Hensbergen. Energy Conservation for Servers. In *Workshop on Power Management for Real-Time and Embedded Systems*, 2001.

[15] T. C. Bressoud and F. B. Schneider. Hypervisor-based fault tolerance. In *Proceedings of the fifteenth ACM symposium on Operating systems principles (SOSP)*, 1995.

[16] J. Chase, D. Anderson, P. Thakur, and A. Vahdat. Managing Energy and Server Resources in Hosting Centers. In *Proceedings of SOSP*, 2001.

[17] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao. Energy-aware Server Provisioning and Load Dispatching for Connection-intensive Internet Services. In *Proceedings of NSDI*, 2008.

[18] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live Migration of Virtual Machines. In *Proceedings of the Symposium on Networked Systems Design and Implementation (NSDI)*, 2005.

[19] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield. Remus: high availability via asynchronous virtual machine replication. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2008.

[20] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini. Coscale: Coordinating cpu and memory system dvfs in server systems. In *Proceedings of the IEEE/ACM International Symposium on Microarchitecture*, 2012.

[21] Facebook Rack-level UPS for improved efficiency. Facebook Rack-level UPS for Improved Efficiency. `http://www.datacenterknowledge.com/archives/2011/04/07/`.

[22] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2007.

[23] X. Fu, X. Wang, and C. Lefurgy. How Much Power Oversubscription Is Safe and Allowed in Data Centers. In *Proceedings of the ACM International Conference on Autonomic Computing (ICAC)*, 2011.

[24] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy. Optimal Power Allocation in Server Farms. In *Proceedings of SIGMETRICS*, 2009.

[25] Google Revenue in 2011. `http://investor.google.com/earnings/2011/index.html`.

[26] S. Govindan, J. Choi, B. Urgaonkar, A. Sivasubramaniam, and A.Baldini. Statistical Profiling-based Techniques for Effective Power Provisioning in Data Centers. In *Proceedings of the ACM European Conference on Computer Systems (EuroSys)*, 2009.

[27] S. Govindan, A. Sivasubramaniam, and B. Urgaonkar. Benefits and Limitations of Tapping into Stored Energy For Datacenters. In *Proceedings of the International Symposium of Computer Architecture (ISCA)*, 2011.

[28] S. Govindan, D. Wang, L. Y. Chen, A. Sivasubramaniam, and B.Urgaonkar. Towards Realizing a Low Cost and Highly Available Datacenter Power Infrastructure. In *Workshop on HotPower*, 2011.

[29] S. Govindan, D. Wang, A. Sivasubramaniam, and B. Urgaonkar. Leveraging stored energy for handling power emergencies in aggressively provisioned datacenters. In *Proceedings of the international conference on Architectural Support for Programming Languages and Operating Systems*, 2012.

[30] J. Hamilton. Internet-scale Service Infrastructure Efficiency, ISCA Keynote, 2009.

[31] JAMES GLANZ. Google Details, and Defends, Its Use of Electricity. `http://www.nytimes.com/2011/09/09/technology/google-details-and-defends-its-use-of-electricity.html?_r=2&`.

[32] A. Kansal, B. Urgaonkar, and S. Govindan. Using dark fiber to displace diesel generators. In *Proceedings of the USENIX conference on Hot Topics in Operating Systems*, HotOS, 2013.

[33] S. T. King, G. W. Dunlap, and P. M. Chen. Debugging operating systems with time-traveling virtual machines. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, 2005.

[34] V. Kontorinis, L. E. Zhang, B. Aksanli, J. Sampson, H. Homayoun, E. P. tis, D. M. Tullsen, and T. S. Rosing. Managing Distributed UPS Energy for Effective Power Capping in Data Centers. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2012.

[35] K. Le, R. Bianchini, M. Martonosi, and T. Nguyen. Cost- and Energy-Aware Load Distribution Across Data Centers. In *Workshop on Power-Aware Computing and Systems (HotPower)*, 2009.

[36] C. Lefurgy, X. Wang, and M. Ware. Server-Level Power Control. In *Proceedings of the International Conference on Autonomic Computing (ICAC)*, 2007.

[37] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan, and C. Kozyrakis. Power management of datacenter workloads

using per-core power gating. *IEEE Comput. Archit. Lett.*, 8 (2), 2009.

[38] D. Linden and T. B. Reddy. *Handbook of Batteries*. McGraw Hill Handbooks, 2002.

[39] M. R. Marty and M. D. Hill. Virtual Hierarchies to Support Server Consolidation. In *Proceedings of ISCA*, 2007.

[40] M. Marwah, P. Maciel, A. Shah, R. Sharma, T. Christian, V. Almeida, C. Araújo, E. Souza, G. Callou, B. Silva, S. Galdino, and J. Pires. Quantifying the Sustainability Impact of Data Center Availability. *SIGMETRICS Performance Evaluation Review*, 37(4), 2010.

[41] D. Meisner and T. F. Wenisch. Dreamweaver: architectural support for deep sleep. In *Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems*, 2012.

[42] D. Meisner, B. T. Gold, and T. F. Wenisch. PowerNap: Eliminating Server Idle Power. In *Proceedings of ASPLOS*, 2009.

[43] D. Meisner, C. M. Sadler, L. A. Barroso, W. Weber, and T. F. Wenisch. Power Management of Online Data-intensive Services. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2011.

[44] MEMCACHED: Memory-object Caching System. `http://www.memcached.org`.

[45] Michael A. Bell. `http://www.it.northwestern.edu/bin/docs/DesignBestPractices_127434.pdf`.

[46] Microsoft Reveals its Specialty Servers, Racks, Apr. 2011. `http://www.datacenterknowledge.com/archives/2011/04/25/microsoft-reveals-its-specialty-servers-racks/`.

[47] J. Moore, J. Chase, P. Ranganathan, and R. Sharma. Making Scheduling Cool: Temperature-Aware Workload Placement in Data Centers. In *Proceedings of USENIX*, 2005.

[48] D. Narayanan and O. Hodson. Whole-system persistence. In *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2012)*, 2012.

[49] R. Nathuji and K. Schwan. VirtualPower: Coordinated Power Management in Virtualized Enterprise Systems. In *Proceedings of the Symposium on Operating Systems Principles (SOSP)*, 2007.

[50] National Survey of Datacenter Outages, 2010. `http://www.inquirere.com/wp-content/uploads/2010/11/National-Survey-on-Data-Center-Outages.pdf`.

[51] S. Osman, D. Subhraveti, G. Su, and J. Nieh. The design and implementation of zap: a system for migrating computing environments. *SIGOPS Oper. Syst. Rev.*, 36(SI), 2002.

[52] E. Pinheiro, R. Bianchini, E.Carrera, and T. Heath. Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems. In *Workshop on COLP*, 2001.

[53] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu. No Power Struggles: Coordinated Multi-level Power Management for the Data Center. In *Proceedings of ASPLOS*, 2008.

[54] P. Ranganathan, P. Leech, D. Irwin, and J. Chase. Ensemble-level Power Management for Dense Blade Servers. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2006.

[55] N. Sharma, S. Barker, D. Irwin, and P. Shenoy. Blink: Managing Server Clusters on Intermittent Power. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2011.

[56] R. Singh, D. Irwin, P. Shenoy, and K. K. Ramakrishnan. Yank: Enabling green data centers to pull the plug. In *Proceedings of 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2013.

[57] SPEC CPU2006. http://www.spec.org/cpu2006/.

[58] SPEC JBB2005: Java Business Benchmark. `http://www.spec.org/jbb2005/`.

[59] J. Stoess, C. Lang, and F. Bellosa. Energy Management for Hypervisor-Based Virtual Machines. In *Usenix Technical Conference*, 2007.

[60] The Cost of Power Disturbances to U.S. Businesses, 2001. `http://www.onpower.com/pdf/EPRICostOfPowerProblems.pdf`.

[61] W. P. Turner, J. H. Seader, V. Renaud, and K. G. Brill. Tier classifications define site infrastructure performance. *Uptime Institute White Paper*, 2008.

[62] A. Verma, G. Dasgupta, T. Kumar, N. Pradipta, and R. Kothari. Server Workload Analysis for Power Minimization Using Consolidation. In *Proceedings of USENIX*, 2009.

[63] D. Wang, C. Ren, A. Sivasubramaniam, B. Urgaonkar, and H. Fathy. Energy storage in datacenters: what, where, and how much? In *Proceedings of the Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2012.

[64] D. Wang, S. Govindan, A. Sivasubramaniam, A. Kansal, J. Liu, and B. Khessib. Underprovisioning backup power infrastructure for datacenters. Technical Report CSE-13-012, The Pennsylvania State University, 2013.

[65] D. Wang, C. Ren, and A. Sivasubramaniam. Virtualizing Power Distribution in Datacenters. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2013.

[66] X. Wang and M. Chen. Cluster-level Feedback Power Control for Performance Optimization. In *Proceedings of HPCA*, 2008.

[67] A. Weisel and F. Bellosa. Process Cruise Control-Event-Driven Clock Scaling for Dynamic Power Management. In *Proceedings of Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, 2002.

[68] Windows Hybrid Low-power Sleep, 2009. `http://windows.microsoft.com/en-us/windows7/sleep-and-hibernation-frequently-asked-questions`.

[69] H. Zeng, C. S. Ellis, A. R. Lebeck, and A. Vahdat. Currentcy: A Unifying Abstraction for Expressing Energy Management Policies. In *Proceedings of the Usenix Annual Technical Conference (USENIX)*, 2003.