The Pennsylvania State University
The Graduate School
College of Engineering

PROVISIONING AND HARNESSING ENERGY STORAGE IN DATACENTERS

A Dissertation in
Computer Science and Engineering
by
Di Wang

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2014

The dissertation of Di Wang was reviewed and approved* by the following:

Anand Sivasubramaniam
Professor of Computer Science and Engineering
Dissertation Adviser, Chair of Committee

Bhuvan Urgaonkar
Associate Professor of Computer Science and Engineering

Chita R. Das
Professor of Computer Science and Engineering

Hosam K. Fathy
Assistant Professor of Mechanical and Nuclear Engineering

Aman Kansal
Senior Researcher, Microsoft Research
Special Member

Lee Coraor
Associate Professor of Computer Science and Engineering
Graduate Director of Computer Science and Engineering

*Signatures are on file in the Graduate School.

Abstract

With terawatts expended in powering today's datacenters, their power consumption poses serious economic, societal and environmental concerns. A large datacenter spends millions of dollars in yearly operational expenditures (op-ex) paying its electricity bills. An even larger capital expenditure (cap-ex) goes into provisioning its power infrastructure, to accommodate the peak power draw, even if this draw is never or rarely sustained. With consumers demanding more for less, extracting the maximum value out of every provisioned and consumed watt in these datacenters is critical to profitability and sustenance.

There have been proposals to underprovision the power infrastructure, sizing it to handle a high percentile of the power draw rather than occasional high peaks. Demand Response (DR) is then employed to address these high peaks at runtime by re-shaping, deferring or modulating demand temporally and spatially. Until now, datacenter DR has primarily used computing knobs such as consolidation, scheduling, migration, and power state modulation. Recently, Energy Storage Devices (ESDs) have been proposed to provide a complementary alternative to these knobs. However, there are multiple issues and challenges in leveraging these ESDs for datacenter DR that require an in-depth study in the datacenter context, which is one major intent of this dissertation.

On the other hand, the primary usage of energy storage in today's datacenters, in the form of Uninterrupted Power Supply (UPS) devices, is to handle power outages. A significant portion of the datacenter power infrastructure's cap-ex goes into the backup power equipment - Diesel Generators (DGs) and UPSes - needed to sustain operation during utility power outages. However, with only a few power outages per year and the majority outages being shorter than a few minutes, it implies opportunities for underprovisioining backup infrastructure for cost savings. 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, which is another major focus of this dissertation.

The goal of this dissertation is to develop solutions for provisioning and harnessing ESDs to (i) improve datacenter demand response capabilities, and (ii) optimize datacenter backup power infrastructure cost via modeling, simulation and experimentation with datacenter workloads. The dissertation consists of two major parts. In the first part, we mainly focus on how to leverage ESDs for enhancing datacenter demand response capabilities. We first develop three different methodologies for provisioning and control energy storage: (1) heuristics based on real datacenter power demand peak and valley characteristics for easy and quick ESD provisioning; (2) an analytical model which captures the important peak/valley attributes and ESD properties to compare different ESD cost-effectiveness in shaving different kinds of power demands; (3) a generalized optimization platform for ESD provisioning, placement and control. Then, we design and implement a system software for managing these ESDs. With a 2-level power hierarchy prototype, we evaluate the impacts and benefits of different mechanisms and policies in harnessing ESDs using datacenter applications. In the second part, we explore the datacenter backup power infrastructure design space, considering cost, availability, performance and application consequences of underprovisioning the backup 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 during a power outage. In the evaluation, we show that Diesel Generators can be completely removed in many cases, and compensated with additional battery energy capacity in the UPS units to achieve desirable cost-performance-availability tradeoff.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Objectives

The growth of Internet-centric services, and increasing reliance on information technology (IT) in organizations has led to proliferation of datacenters housing IT resources across these enterprises. There is rapid growth of such datacenters in numbers as well as in size. This growth is raising concerns regarding their cost, environmental footprint and scalability. Power consumption of datacenters has tremendous implications on these concerns. A large datacenter spends millions of dollars in yearly operational expenditures (op-ex) paying its electricity bills. An even larger capital expenditure (cap-ex) goes into provisioning its power infrastructure as shown in Figure 1.1, to perform power conversion, accommodate the peak power delivery (even if this draw is never or rarely sustained), guarantee power quality, and backup power during utility outages (even outages are rare and mostly short if happened). The cost of powering the nation's datacenters is expected to exceed $15 billion over the next decade [60] , imposing a peak load of over 20 GW on the grid. Peak load correlates to number of required power plants. Each 100 MW power plant costs $60-100 million to build, emitting over 50 million tons of CO2 over its lifespan [32]. Further, power is often a limiting factor in the scalability of datacenters [87, 14, 80, 21, 55, 56], because of: (i) the limits - hard or soft (pricing) - in the draw imposed by utilities (since avoiding black/brown outs when demand exceeds supply may mandate additional power plants), which is reflected in the

op-ex. The peak IT demands also coincidentally occur during peak tariff periods; and (ii) the cap imposed by the power distribution infrastructure within the datacenter which constitutes the cap-ex (studies [9, 50, 12] have estimated $10-20 for each watt of power infrastructure provisioning, even if this watt is not consumed). It is thus essential to extract the maximum value out of every *provisioned* and *consumed* watt in the datacenter. Towards this purpose, this dissertation focuses on two main aspects to address these concerns.

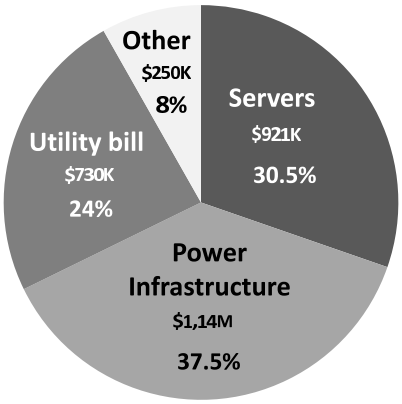
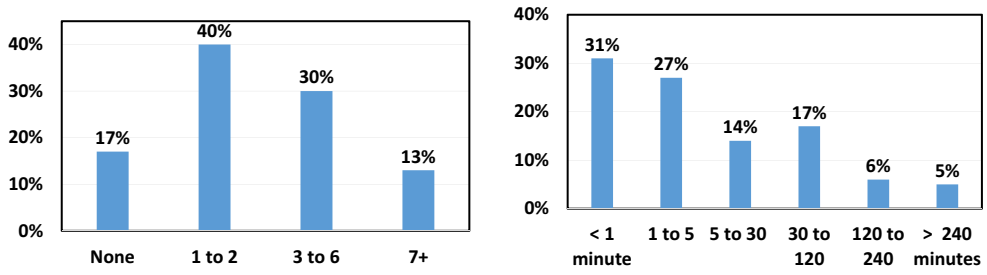


Figure 1.1: Amortized *Monthly* Costs for Physical Infrastructure of a 10 MW Datacenter. Source: [50, 9, 85].

Energy Storage for Demand Response: Recently, there have been proposals [35, 48, 59, 113, 40] to underprovision datacenter power infrastructure, sizing it to handle a high percentile of the power draw rather than the occasional high peaks. Demand Response/Shaping is a common technique (as in electricity grids) to match and shape the consumption/demand to the supply vagaries - capacity, costs, intermittency, quality, etc. Datacenter demand-response (DR) primarily uses IT-based knobs - consolidation [20, 88, 105, 42, 68, 43, 28], scheduling [80, 22], migration [45, 42, 61, 22], and power state modulation [89, 114, 41, 15, 72]- to shape power draw. One common mechanism - energy storage - that is used in grids (and other domains) to hoard power when available and draw from it when needed (to address supply-demand mismatch) has only recently drawn attention for datacenter DR [46, 48, 103, 59, 93, 49]. Energy storage in datacenters has been primarily used as a transition mechanism to temporarily handle power

outages until diesel generators are brought on-line, and not for demand response solutions. We believe that *Energy Storage Devices (ESDs)* can be a valuable resource for demand response in a datacenter to reduce both cap-ex and op-ex costs. Consequently, there is a rich set of research issues relating to provisioning, integrating and harnessing these devices in the datacenter context to reduce power related costs, which is one focus of this dissertation.

Energy Storage for Power Outages: 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% [77]) goes into the backup power equipment - Diesel Generators (DGs) and Uninterrupted Power Supplies (UPS) - needed to sustain operation during utility power outages. 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 dissertation, we investigate the relatively unexplored area of underprovisioning, and perhaps even removing, parts of the backup power infrastructure.



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

Figure 1.2: Power Outages Distribution for U.S. Business (Source: [101, 83])

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 1.2 shows the distribution of power outages in a typical year for US businesses [101, 83]. In Figure 1.2(a), we observe that 6 or fewer outages are the overwhelming majority (in 87% of the businesses). Further, in Figure 1.2(b) we see

that a large majority (over 58%) of these outages are 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 re-direction/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 [53]). 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?* Another focus of this dissertation 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.

This dissertation develops solutions for provisioning and harnessing ESDs to (i) improve datacenter power demand response capabilities, and (ii) optimize datacenter backup power infrastructure cost via modeling, simulation and experimentation with datacenter workloads. It investigates issues on provisioning and operating a diverse set of ESDs (different kinds of batteries, ultracapacitors, flywheels, compressed air.) in the datacenter, while customizing their usage based on datacenter workloads. It develops system software support for managing these devices as a first class resource. Moreover, it explores the design space of backup power infrastructure, and evaluate various system techniques for under-provisioned backup infrastructure (i.e., completely remove Diesel Generators) by procuring more energy storage capacity to achieve desirable cost-performance-availability trade-offs.

1.2 Contributions

This dissertation makes the following specific research contributions.

Methodology Development for Provisioning and Controlling Energy Storage (Section 3.1).

- Heuristics based on Peak and Valley Attributes and Their Correlations (Section 3.1.1): We abstract power demands in the form of peaks and valleys, identify attributes for peaks and valleys, and important correlations across these attributes that can influence the choice, provision and control effectiveness of ESDs for power demand modulation. We characterize these attributes and their correlations with real datacenter power traces and come up with simple provisioning heuristics and control implications.
- Analytical Model (Section 3.1.2): We develop an analytical model to compare different ESD technology cost-efficacy (provisioning cost) in shaping different kinds of power demands. The analytical model considers both workload properties (peak and valley attributes) and ESD characteristics such as power and energy cost, charge and discharge rate, life cycles and depth of discharge.
- Optimization Framework (Section 3.1.3): We formalize a systematic framework for capturing the different intricacies of multiple kinds of ESD technologies (including energy efficiency, health degradation, self-discharge, energy and power densities) as well as the possibility of having them at multiple locations within a datacenter, develop a generalized optimization platform to jointly address ESD provisioning, placement and control problems in datacenters.

System Software for Managing Energy Storage (Section 3.2).

- Virtualize datacenter power hierarchy (Section 3.2.1): A software-based virtual power hierarchy (vPower) (from the datacenter at the root, to the servers) for each application, within which it can safely execute, insulated from any power-related emergencies arising from coexisting applications. The power hierarchy that is being virtualized includes the capacities of ESDs in each layer as well as all the power equipments.
- Design, Implement and Evaluate System Software Mechanisms and Policies (Section 3.2.2, 3.2.3): On the mechanisms front, we explore the feasibility and benefits for (i) applications to explicitly specify their power needs, at different temporal and accuracy levels; and (ii) systems software to explicitly manage the power hierarchy (sourcing from an ESD, gauging the draw in different portions of the hierarchy, etc.), apart from traditional power state/DVFS control knobs. On the policies front, we investigate (i) admitting and colocating the right set of applications under the same constrained hierarchy; (ii) accounting for their power usage (iii) policing application power draw to the allocation and ensure fairness; and (iv) leveraging the complemen-

tary interactions between ESDs and different computing knobs.

Underprovisioning Backup Power Infrastructure (Chapter 4).

- Design Space Exploration for Backup Infrastrucutre (Section 4.1): 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.
- Evaluate System Techniques for Handling Outages with an Underprovisioned Backup Infrastructure (Section 4.2, 4.3): We observe several interesting insights based on our evaluation leveraging software and hardware techniques for diverse datacenter applications, subjected to a variety of power outage duration. We find 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 find 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 todays approach.

1.3 Organization

The remainder of this document is organized as follows. First, different ESD technologies and their characteristics are introduced in Chapter 2. Chapter 3 presents methodologies for provisioning these ESDs, and system software for managing ESDs for datacenter power demand response. In Chapter 4, we study the design space trade-offs when building different backup capacity alternatives for a datacenter, and explore using energy storage to underprovision backup infrastructure especially diesel generators. Chapter 5 discusses related work. Finally, we summarize our work and discuss future directions in Chapter 6.

Chapter 2

Energy Storage Devices (ESDs)

In this chapter, we present necessary background of ESD in datacenters.

Current Primary Role of ESDs in Datacenters: Energy storage in datacenters has been primarily used as a back-up mechanism to temporarily handle power outages until diesel generators are brought on-line. This typically takes a few seconds (at most a couple of minutes), and most datacenters provision UPS devices with lead-acid batteries of possibly 2X-3X this capacity. Many datacenters also have multiple (e.g., N+1 redundancy) UPS units for enhancing availability.

ESDs to Lower Power Costs: Recently, some proposals have been made to leverage either these devices, or add additional storage capacities, for DR within a datacenter [46, 48, 103, 59] to reduce power related cap-ex and op-ex. The cap-ex of provisioning the power infrastructure is reported to cost between $10-20 per watt [9, 50]. For safety reasons, it needs to be provisioned for the peak draw that may ever happen (even if it is a rare event). Shaving peak draws can, therefore, reduce cap-ex costs. Furthermore, shaving at the bottom/lower layers of the hierarchy benefits a larger portion of the hierarchy compared to shaving only at the top level - shaving at a level allows all the equipment in that level and above to be provisioned for a smaller peak. The op-ex that goes into paying the monthly power bills can have different tariff structures depending on the utility. Since peak draws at the utility scale are bound by the capacity of the grid and the generation capacity of power plants, utility companies dis-incentivize high peak draw with either (i) time-of-day pricing [18] (high prices occur when everyone is expected to have high draws), and/or (ii) peak slab based pricing [30] (where the maximum draw is separately tracked and priced for each watt of this draw). Thus, reducing

the peak draw can also reduce op-ex. ESDs can hoard energy during a ”valley” (period of low power draw), which can then be used to supplement the utility when a peak occurs - in effect, hiding a portion of the peak from higher up in the hierarchy (and the utility) - to provide both cap-ex and op-ex savings.

Centralized vs. Distributed ESDs: ESDs, while continuing their role in handling outages, can be introduced at each level of the hierarchy. Most current datacenters use centralized UPS units, i.e., only at the datacenter level, which are typically lead-acid based with enough capacity to sustain the datacenter needs for a few minutes. Newer high-density datacenters are starting to avoid centralized UPS, and are going for decentralized options, e.g., rack-level in Facebook [33], Microsoft [79], or server-level as in Google [44]. Whether centralized or distributed, all these are still single-level ESD placement options. We could have ESDs (possibly different technologies) simultaneously present at different layers hierarchically.

We note the following qualitative comparisons between centralized vs. distributed/hierarchical placement options:

- Distributed ESD placement, particularly deep down (e.g., server level), allows cap-ex savings in a larger portion of the hierarchy.
- There is an analogy with issues in shared vs. private caches in chip-multiprocessors. While a shared (centralized ESD) cache allows better resource utilization when there is imbalance across different users, resource contention can become a problem leading to potential unfairness. Private (distributed ESDs) caches are isolated from each other from this perspective. For example, two server-level ESDs that can each shave a single peak, could do so for their respective servers. It is possible that when replacing them with a single ESD at the higher level (with sum of their capacities), this ESD ends up shaving two peaks of 1 server and none of the other (becoming unfair).
- The statistical multiplexing effect as we move up the datacenter power hierarchy can be in favour of centralized placement given less degree of burstiness at the centralized level and hence smaller capacity is needed from ESDs provisioned at this level for shaving peaks.
- As noted earlier, a completely distributed server-level ESD option can help avoid double-conversion related energy losses compared to centralized placement.
- On the other hand, a centralized ESD solution may not have a serious volume/real estate constraint. In fact, many facilities place such ESDs outside of the datacenter

itself (e.g., separate room of shelves with lead-acid batteries, flywheels, basements of buildings for compressed air, etc.), since datacenter floor/rack space is precious (running to thousands of dollars per square foot). A distributed/hierarchical solution would need real-estate within the datacenter.

- Finally, from the maintenance perspective, distributed/hierarchical ESDs may require much more maintenance effort/cost (e.g., ESD replacement due to failures, end of lifetime, etc) compared to centralized placement where maintenance can be relatively easy.

Representative ESD Technologies: Significant advances have been made in energy storage technologies, both in extending the practicality of their applications (by boosting densities, increasing lifetimes, lowering cost, etc.), as well as in bringing new technologies to the market. Currently, there is a broad spectrum of energy storage options exists, as is typically depicted using Ragone plots [90], such as the one shown in Figure 2.1. A Ragone plot compares different ESD technologies in terms of their power densities and energy densities, where ”density” can be either volumetric or mass-based. As can be seen in Figure 2.1, there are sharp differences between various ESDs. For instance, compressed air-based energy storage (CAES), has a relatively high specific energy but a low specific power, implying that it is better suited for holding a large amount of energy as long as this energy does not need to be discharged very fast. (i.e., long power draws as opposed to short and bursty power draws). On the other hand, capacitors have high power densities, albeit they are unable to not as efficient in sustain a large power draw for an extended period of time. In addition to this broad characterization, there are numerous other factors that can impact ESD suitability for DR, which we discuss below. is elaborated in this section.

In this dissertation, we use the following representative ESD technologies (which are also gradually finding their way into the datacenter [71]) from the Ragone design space in our discussions:

- *Ultra/Super-capacitors (UC):* These improve on conventional capacitors using double-layer electrochemistry between the electrodes, allowing a thousand-fold increase in energy stored. While the double-layer restricts it to low voltage operation, these capacitors can be connected in series for higher voltage needs. There has been nearly two orders of magnitude improvement in their cost per Farad over the past decade, and we are already seeing some commercial offerings of these products [70] for the

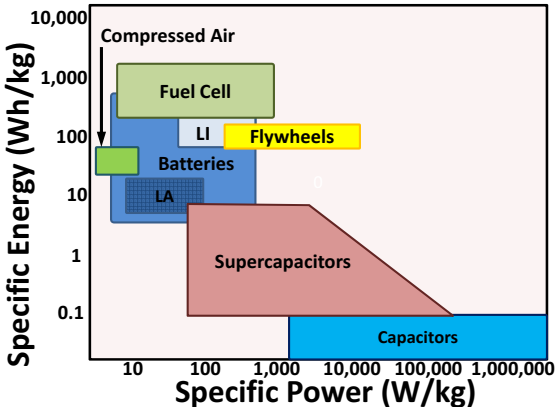


Figure 2.1: Ragone Plot.

datacenter market as UPS devices. We use this ESD as representative of the lower right end of the Ragone plot.

- *Compressed Air Energy Storage (CAES):* Air can be compressed (with compressors which consume energy) in confined spaces, and this pressurized air can subsequently be used to drive turbines for electricity generation. Since compression and decompression generate and absorb heat respectively, heat exchange mechanisms are needed to ensure proper operation. More importantly, CAES [23] may require significant real-estate to create such confined spaces, and the cost/availability of such real-estate needs to be taken into consideration (e.g., building basements, tanks in parking lots, etc.). We use CAES as representative of the upper left end of the Ragone plot.
- *Flywheels (FW):* The momentum of a rotating wheel/cylinder is gaining acceptance [107] as a UPS device for the datacenter, to temporarily handle the load until diesel generators kick in. Even though they are not intended for extended operation (somewhat inferior to even batteries from this perspective), they can provide the high power needs in the brief interlude between the outage and generator start-up.
- *Batteries (LA or LI):* These (particularly lead-acid) are the most commonly used storage devices in datacenters, where the electrochemical reactions (of the appropriate chemistry within), is used to store and generate electricity. There are several kinds of

batteries [29] - lead-acid, lead-carbon, lithium-ion, vanadium flow, sodium-sulphur, etc., and we will consider lead-acid (LA) and lithium-ion (LI), which are more prevalent and representative of two ends of battery spectrum on the Ragone plot, and with very different costs, in our evaluations.

ESD	LA	LI	UC	FW	CAES
Energy Cost C_k^{eng} ($/kWh)	200	525	10000	5000	50
Power Cost C_k^{pow} ($/kW)	125	175	100	250	600
Energy Density v_k^{eng} (Wh/L)	80	150	30	80	6
Power Density v_k^{pow} (W/L)	128	450	3000	1600	0.5
Discharge:Charge Rate γ_k	10	5	1	1	4
Life Cycle $Lcyc_k$ (# discharges x 1000)	2	5	1000	200	15
Max. DoD DoD_k^{max} (%)	80	80	100	100	100
Float Life T_k^{max} (years)	4	8	12	12	12
Energy Efficiency η_k (%)	75%	85%	95%	95%	68%
Self-discharge μ_k per Day	0.3%	0.1%	20%	100%	low
Ramp Time T_k^{ramp} (sec)	0.001	0.001	0.001	0.001	600

Table 2.1: ESD parameter values [23, 29, 92, 94].

ESD Characterisitics: In addition to this broad characterization, there are numerous other factors that can impact ESD suitability for DR. In Table 2.1, we quantify relevant parameters for the 5 technologies that we evaluate.

- *Cost (Energy and Power):* When optimizing electricity costs with ESDs, we need to account for the costs of the ESDs themselves. The cost of ESD depends on 2 factors - the total energy that is to be stored/discharged, and the rate (power) at which the energy is to be charged/discharged. This is somewhat indicated by where the device

falls on the Ragone plot. Rows 1 and 2 of Table2.1 show these two components of the cost for the 5 ESDs under consideration. As can be expected, from the energy point of view, CAES is the least expensive (50 $/kWh) with ultra-capacitors at the other end of the spectrum at 200X this cost. On the other hand, with respect to power, ultra-capacitors are the most attractive option with CAES turning out to be being 6X more expensive. Batteries offer a good compromise between these extremes. from these perspectives.

- *Density (Energy and Power):* Beyond costs, it is also important to consider the densities of these technologies required to provide a certain energy and power demand. Density determines the "volume" (real-estate) that needs to be provisioned in the datacenter to sustain the demands. Since datacenter real-estate is very precious - whether it be rack space or floor space, volume constraints may need to be imposed when provisioning ESDs. ESDs which may be attractive based on energy or power costs may not necessarily be suitable because of space constraints. For instance, CAES - the most cost-attractive option - is the worst from the density viewpoint even if we are only trying to cater to energy demands (and willing to tolerate the slow discharge rate offered by CAES). In fact, as we will find, trying to provide CAES for each server (or even a rack), is prohibitive in real-estate demands. At best, we can consider CAES at a datacenter scale, where basements, sealed tanks in parking lots, etc., may be options.
- *Discharge/Charge Rate Ratio:* Since ESDs alternate between charging and discharging, it is important that there be sufficient charging time to hoard the required capacity before the next discharge. We can capture this by the discharge/charge ratio, which is larger than 1 (i.e., it takes longer to charge than discharge) for many ESDs. Ultra-capacitors and flywheels may come close to this ideal behavior, while batteries (particularly LA) are not as attractive.
- *Replacement Costs (Charge/Discharge cycles and Lifetime):* We need to consider the costs of ESD replacement, since the datacenter infrastructure may itself have a much longer lifetime (e.g., 12 years as suggested in [50]). The lifetime of an ESD, especially batteries, depends (among other factors) on the number of charge-discharge cycles and the Depth-of-Discharge (DoD) of each discharge [23]. In addition, the internal chemistry itself has certain properties such as lead-out, which can also impact the lifetime orthogonal to usage. Row 8 of Table 2.1 gives the average lifetime (in years) of these ESDs based on typical usage. Batteries typically need replacement

while our other technologies can possibly match the expected datacenter lifetime. It is not that batteries stop working abruptly - rather, their capacity for holding charge degrades over time, and replacement is done when it drops below 80% of the original capacity.

- *Energy Efficiency:* The energy used to charge an ESD is higher than what can be drawn out subsequently, implying losses. Ultra-capacitors and flywheels are very energy efficient, while batteries can incur losses of 15-25% based on their chemistries. The efficiency of CAES is even worse.
- *Self-Discharge Losses:* ESDs can lose charge even when they are not being discharged, with the loss proportional to the time since the last charge. Fly-wheels can be poor from this perspective, and so are ultra-capacitors. Consequently, it may be desirable that such devices be charged just before a discharge, rather than hoarding the charge for a long time.
- *Ramp Rate:* While power density is one factor influencing the rate at which energy can be drawn, the ramp rate is another consideration in some ESDs. One can view the ramp rate as a start up latency to change the power output (analogous to how combustion engines of automobiles can accelerate from zero to a given speed in a certain amount of time, and then sustain it at that speed). In most ESDs we consider the ramp rate is very high (i.e., it takes at most a few milliseconds to start supplying requisite power draw), except in CAES where it can take several minutes. Hence, CAES cannot instantaneously start sustaining any desired draw, requiring either (i) anticipating the draw and taking pro-active measures, or (ii) using some other ESD until CAES becomes ready to sustain the draw.

Given a plethora of ESD options and the idiosyncracies of each option in cost, density, wear, and operational characteristics, there are several interesting research questions - which storage devices? how much capacity? where do we place them? when to charge and discharge? - that we will develop methodologies to address in the next chapter.

Chapter 3

Energy Storage for Enhancing Power Demand Response

3.1 Methodology for Provisioning and Controlling Energy Storage

To address issues on provisioning, placing and operating a diverse set of ESDs in a datacenter leveraging their capabilities and customizing their usage based on datacenter workloads, we develop three different methodologies, each with its own pros and cons on (i) the issues it can solve (e.g., provisioning, placement, control, etc.), (ii) workload information requirement (e.g., whether the entire trace or statistical properties from the workload) and (iii) ESD properties it can capture (e.g., power/energy cost, density, state of health, energy efficiency, etc.).

3.1.1 Heuristics Based on Trace Characterization

As a first step, we start with real datacenter power consumption traces for developing ESD provision and control heuristics. By abstracting power trace in forms of peaks and valleys and characterizing their attributes and correlations, we come up with rough estimates for ESD technology, capacity provisioning and control decision implications. The peak and valley statistical properties enable an easier analysis to quickly examine ESD provision and control issues compared to a time consuming design and evaluation

loop that may require access to the fine-resolution data over an extensive period of time.

Abstracting Peaks and Valleys We first characterize and analyze power demand time-series into a convenient set of abstractions that facilitate ESD provisioning and control. There is a spectrum of abstractions possible, ranging from the very detailed, such as a spatio-temporal reproduction of the entire power demand data at fine resolutions, to a very succinct and possibly simplistic statistic, such as a CDF used in [35]. Our goal is to define and characterize those attributes which would really impact the provision and control of energy storage for power capping. Towards this goal, we propose to abstract the power data using power *peaks* and *valleys* that are based on a specified power cap, and identify different attributes for these peaks and valleys.

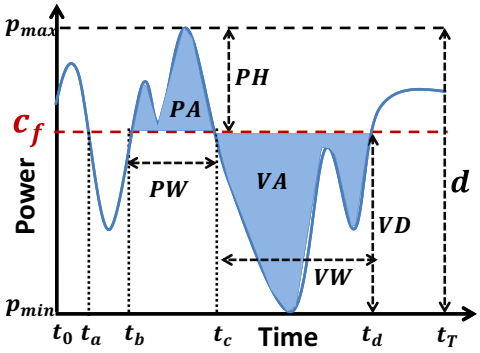


Figure 3.1: Notation for peaks, valleys, and their attributes.

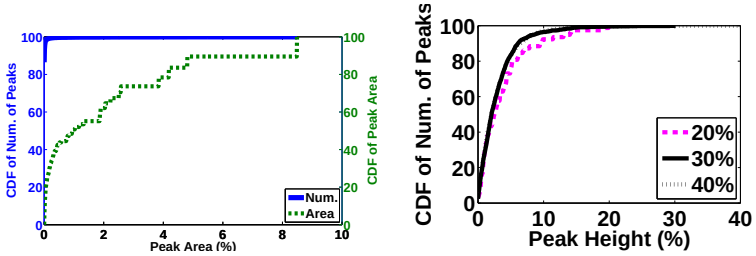
Consider an IT equipment power demand time series p_t, over discretized time $t = 0...T$, in time steps of Δt. This demand could be either from a single server, or a rack, a cluster, or even a datacenter. Let p_{min} and p_{max} be the minimum and maximum power demand over all t in this time series. The dynamic range of power consumption, denoted d, is given by $d = p_{max} - p_{min}$. Consequently, we define a power cap c_f, in terms of the fraction, f, (percentage) of this dynamic range. The absolute power cap is then given by $c_f = (1 - f) \times d + p_{min}$. Setting a cap of c_f to the time series p_t gives rise to peaks and valleys in the power draw as illustrated in Figure 3.1. For instance, the power draws in the time intervals $[t_0, t_a]$ and $[t_b, t_c]$ are example peaks, while the power draws in time intervals $[t_a, t_b]$ and $[t_c, t_d]$ are example valleys in this figure. Formally, we can define a series $\{i_1...i_k\}$ of points in time where the power demand intersects the horizontal line at

a given c_f, i.e. $\forall t \in i_k, p_t = c_f$. Note that the power demand in any interval $[i_k, i_{k+1}]$ can be categorized as either a peak or a valley. It is a peak if the power demand within this interval exceeds c_f, and is a valley otherwise. In addition, we also need to consider the extreme cases of intervals $[t_0, i_1]$ and $[i_k, t_T]$ where the beginning and end of time series do not intersect with the horizontal line c_f. These intervals can also be categorized as a peak or a valley depending on whether the power demand in those intervals fall above or below c_f respectively.

The power demand time series p_t, can now be expressed as a sequence of peaks and valleys of different intervals defined by their respective $[i_k, i_{k+1}]$, with k used to denote the index in the sequence. An interval k, whether a peak or a valley, can be characterized by the following attributes, each of which can have an implication on power capping:

- *Peak Height (PH_k) or Valley Depth (VD_k):* When k is a peak, its height (Power) can be specified as $PH_k = \frac{max_{i_k \leq t \leq i_{k+1}}\{p_t\} - c_f}{d}$. This is the maximum power draw exceeding the defined cap that needs to be provided over the duration of this peak, normalized as a fraction or percentage of the dynamic power range d. The magnitude of PH_k would determine the capacity of an energy storage device to sustain this peak power need. Similarly, when k is a valley, its depth can be specified as $VD_k = \frac{c_f - min_{i_k \leq t \leq i_{k+1}}\{p_t\}}{d}$. This is the lowest power draw during this valley, capturing its ability to re-charge an energy storage device.
- *Peak Width (PW_k) or Valley Width (VW_k):* This is simply the duration (time) of the corresponding peak or valley and is calculated as $i_{k+1} - i_k$. Valley width corresponds to the inter-peak time and vice-versa. These attributes show the frequency of their occurrences, thereby giving an indication of recovery periods between peaks.
- *Peak Area (PA_k) or Valley Area (VA_k):* The area of a peak area PA_k (Energy) is given by $\sum_{t=i_k}^{t=i_{k+1}} (p_t - c_f) \times \Delta t$. Correspondingly, the valley area VA_k is given by $\sum_{t=i_k}^{t=i_{k+1}} (c_f - p_t) \times \Delta t$. The peak area corresponds to the total energy exceeding the power cap, thereby indicating the amount of work that needs to be accommodated by ESDs without the help of the additional power. The valley area is indicative of how much extra work can be accommodated within the specified cap, e.g. amount of energy for re-charging a storage device.

Heuristics based on Characteristics We collect fine-grained (20 seconds resolution) power demand data from Microsoft production servers and characterize its peak and



(a) Peak Height Distribution (b) Peak Area Distribution

Figure 3.2: Peak Height and Area Characteristics

valley attributes and their correlations. We then develop heuristics based on these characteristics for quantifying ESD capacity and dispatching ESDs.

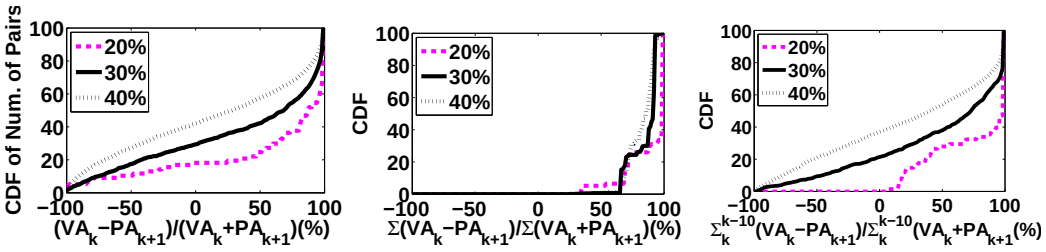
Quantifying Capacity: The shape - height which indicates peak power, and area which indicates energy needs - of the peaks determines the required ESD capacity for any given technology. We, thus, examine the characteristics in Figure 3.2 (a) and Figure 3.2 (b). Rather than go for the 100-th percentile, we pick the knees of these curves - 90th percentile of the peaks in terms of height, peaks contributing to the 90% of total area under peaks - with the former indicating the power capacity and the latter representing the required energy capacity. The maximum between these two is what needs to be provisioned. However, we also need to ensure that there is sufficient slack in the valleys to re-charge for this capacity. For this purpose, we examine Figure 3.3 to examine the re-charging opportunities. While the immediately preceding valley(s) may not have enough slack for such re-charge (Figures 3.3 (a) and (c)), the results in Figure 3.3 (b) suggests that greedily re-charging at every opportunity may provide the slack to re-charge for this capacity to suppress all peaks.

Using these rules-of-thumb as a heuristic for ESD provisioning, in Table 3.1 we show the capacity selected by this heuristic for one specific ESD technologies - lead-acid battery, showing its effectiveness in shaving the peaks (both number and area). We compare these results with an Optimal capacity provisioning algorithm, that is guaranteed to provide the minimal capacity to shave all peaks in the given data. While the latter does need to extensively run through the time series of power demands, to ensure these guarantees (minimal capacity, charge/discharge rate guarantees, account for energy losses, etc.), we find that our simple heuristic approach shaves over 99% of the peaks, and nearly 90% of the peak area, with a capacity (and corresponding cost), that

is less than half of the capacity (for lead-acid batteries) than what the Optimal algorithm specifies.

Tech.	Approach	Capacity (% peak area)	Cost (% of Opt LA)	Peaks shaved (% of total peaks)	Area shaved (% of total peak area)
LA	Heuristic	6.0	47	99.97	89.42
LA	Opt.	12.1	100	100	100

Table 3.1: ESD capacity provisioning for Lead Acid (LA) battery.



(a) Immediate Preced. Valley

(b) All valleys till till that peak

(c) Preceding 10 Valleys

Figure 3.3: Peak and Preceding Valley(s)

Control Implications: The correlations between a peak and its preceding valleys is important for ESD-based peak suppression, which relies on previous valleys to re-charge its capacity for sourcing power during the current peak. We mainly look at the area (energy) capacity of the valleys for charging opportunities.

Figure 3.3 (a) captures the area difference of a peak and its preceding valley. With 20% caps, nearly 80% of the preceding valleys have sufficient energy charging capacities for the following peak. This percentage decreases to 60% for the 40% cap. Figure 3.3 (b) captures the opportunities for re-charging in all previous valleys, while still discharging for all previous peaks, by showing the CDF of cumulative (from 1 to $k+1$) area difference between each peak and its preceding valley of the entire trace. We note that there are absolutely no negative values in the CDF, indicating that a sufficiently-sized ESD which uses every valley to re-charge, and discharges for every peak, will never run out of capacity to shave any peak in the entire execution. Consequently, these

characteristics show a lot of promise for ESD-based peak suppression from a theoretical perspective. However, practically there are several conditions limiting ESD charge and discharge opportunities. Hence, we also look at a limited window of opportunity for re-charging, by considering the area differences of 10 consecutive valley-peak pairs in Figure 3.3 (c). While the 20% power cap imposition can be completely met by the ESDs with this limited window, the results for the 30% and 40% caps are still showing significant negative values. The promise of a much larger window (as in Figure 3.3 (b)) suggests that an ESD based solution should also look to aggregately suppress these bursty peaks, rather than just-in-time charge and discharge based solutions.

While simplistic, this approach allows us to prune out some obviously undesirable solutions in the vast design space and give reasonable estimations. However, this approach relies on analysis of detailed power traces which may change overtime and are not always available. Moreover, it ignores many facets of ESD operation such as lifetime/reliability, conversion efficiency, etc, and lacks of detail charging/discharging decisions, which may lead to misleading cost-efficacy analysis.

3.1.2 Analytical Model

Towards the downside of the previous strategy, we intentionally keep the power demand representation simplistic and capture ESD characteristics with a simple analytical model.

The power demand is simply considerred as an ON-OFF series where the ON periods correspond to a high demand value ("peaks") while the OFF periods correspond to a low demand value ("valleys"). Furthermore, we denote the mean amplitude ("height") and duration ("width") of the peaks and valleys of this time-series as are chosen from a normal distribution with mean values (h_{peak},w_{peak}) and (h_{valley},w_{valley}), respectively (the variances are intentionally kept small to glean more insights at this stage). We focus on finding its size/cost that is needed to "shave" a certain specified portion of *all* the peaks (rather than any one peak) in the time-series.

Model for a single ESD To achieve effective selection and placement of ESDs in the datacenter, it is important to understand the efficacy of each ESD technology in shaping a given power demand time-series, and use this to understand the trade-offs across these

technologies.

We use the same notations for peak attributes as in Section 3.1.1. We denote the "frequency" of peak occurrences as $f_{peak} = \frac{1}{PW+VW}$. We use $k \in \{1, 2, ..., K\}$, to denote an ESD technology (so $K = 5$ in our evaluations). We can now translate the ESD properties into the following constraints to calculate the cost of ESD k (C_k^{total}) that must be provisioned to shave PH from the demand.

First, the power and energy densities of technology k impose these lower bounds on the required ESD capacity (cost):

$$\frac{C_k^{total}}{C_k^{pow}} \geq PH; \quad \frac{C_k^{total}}{C_k^{eng}} \times DoD_k^{max} \geq PH \times PW.$$

Next, the device must possess enough energy at the beginning of a peak to shave PH of it. It may have to acquire all of this energy by re-charging during the preceding valley, implying the following dependence on the discharge/charge rate ratio (γ_k):

$$\frac{C_k^{total}}{C_k^{pow}} \times VW \times \frac{1}{\gamma_k} \geq PH \times PW.$$

Finally, the cost of ESD replacement should also be factored. This is governed by its expected float-life (T_k^{max}), as well as (i) the wear caused by repeated discharges and (ii) the extent/depth of each these discharges. Except for batteries, the other 3 ESDs have a large enough T_k^{max} to be less affected. To quantify the effect of wear on the lifetime, we can de-rate the expected number of lifetime charge-discharge cycles ($Lcyc_k$) which is calculated pessimistically at a Depth-of-Discharge of DoD_k^{max}, by the actual depth to which it is discharged DoD_k^{actual}, to get the expected lifetime of the ESD k as

$$Life_k = min\left(\frac{1}{f_{peak}} \times Lcyc_k \times \frac{DoD_k^{max}}{DoD_k^{actual}}, T_k^{max}\right).$$

This expected lifetime can be used to find the amortized cost (yearly) as per each of the above three requirements (energy needs, power needs, re-charging rates), and the cost of the employed ESD k is given by the maximum of these three requirements:

$$max\left(\frac{PH \times C_k^{pow}}{Life_k}, \frac{PH \times PW \times C_k^{eng}}{Life_k \times DoD_k^{max}}, \frac{PH \times PW \times \gamma_k C_k^{pow}}{VW \times Life_k}\right).$$

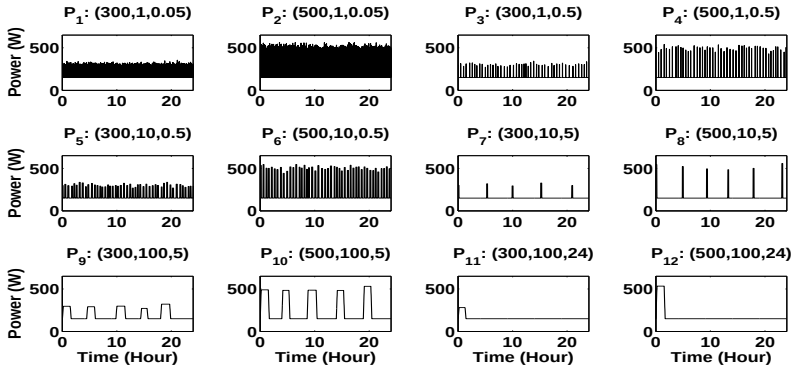


Figure 3.4: Power demand building blocks with mean values of (PH watt, PW min, $\frac{1}{f_{peak}}$ hour)

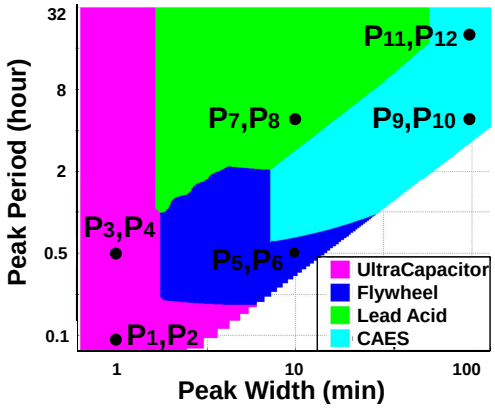


Figure 3.5: Most cost-effective ESD for different Peak types.

ESD Suitability for Different Demands Since at this stage our goal is to evaluate the match between ESD properties and workload characteristics, we begin with synthetic power demands (using our ON-OFF series described above) for which we vary the parameters over a wide range. We use normal distributions for each of these parameters with specified mean and variance. We pick a set of 12 different combinations of means (2 each for peak height and frequency, 3 for peak width, and 1 for valley height) that we use as our synthetic workload "building blocks". Figures 3.4 (P_1-P_{12}) shows the 12 resulting power demands. For each of these 12 power demands, we choose an PH that is realizable for that demand.

Using this model, we compute the (peak width, peak frequency) region over which

each of the 5 ESDs under consideration is the most cost-effective (since PH is set to be the same for all ESDs, the resulting cap-ex and op-ex savings are identical, and hence we need only compare the ESD costs). We show these regions in Figure 3.5 and we identify three main insights.

No Single Technology Always Best: For each storage technology, there is a portion of the workload region, where it is cost-superior to other technologies. For example, ultra-capacitor is best when we have extremely narrow peaks (tens of seconds to a minute) as in P_1-P_4. Among the 5 technologies, ultra-capacitors offer the cheapest cost/power draw ($/kW). They can also re-charge fast enough within the high frequency of peak occurrence in these demands. Although they are the most expensive in terms of cost/unit energy ($/kWh), this does not become prohibitive since the peaks to be shaved for these demands require only small amounts of energy. At the other end, CAES is an attractive option for demands P_9-P_{12}, which are more "energy-demanding" with wide peaks (and ultra-capacitors are high cost options for these demands). CAES, on the other hand, requires very high capacities (and costs) to handle the narrow high power peaks where ultra-capacitors are attractive. There are regions in the middle where batteries (P_7-P_8) and flywheels (P_5-P_6) are the better options. Therefore, a datacenter may need to consider these workload idiosyncrasies and variances over its lifetime in provisioning ESDs rather than always employing a fixed technology.

Hybrid ESDs May Be Desirable: Certain technologies appear complementary to each other in terms of their pros and cons. The most stark contrast is between ultra-capacitors and CAES: while ultra-capacitors are the most cost-effective for P_1-P_4, CAES turns out to be the most prohibitive; the reverse holds for P_9-P_{12}. Such complementary behavior is also seen, to different degrees, for other pairs of technologies as well. Therefore, when a datacenter may house different kinds of workloads - either across its different spatial regions or temporally over its lifetime - it may be worthwhile to consider hybrid ESD options.

Multi-level ESD May Be Desirable: Within a datacenter, the nature of the power demand seen at different levels of its power hierarchy can be different, e.g., higher averages and smaller variances because of statistical multiplexing effects as we move up. When we compare a power demand with smaller average but higher variance (e.g, P_1) against those with higher average but smaller variance (e.g., P_{10}), we find different technologies being the most cost-effective. Whereas P_1 is best shaped using ultra-capacitors, CAES

is the best choice for P_{10}. Therefore, it may be desirable to employ appropriate (possibly different) ESD technologies at multiple - server, rack, and datacenter - levels of the power hierarchy. Furthermore, pushing ESDs deeper down the hierarchy can allow higher cap-ex savings.

3.1.3 Optimization Framework

Motivated by the insights from previous analytical model and limitations of the model being only able to analyze homogeneous ESDs and periodic peaks/valleys, we then develop a generalized optimization platform for ESD provisioning, placement and control in the datacenter. In particular, we design it to allow for multiple kinds of ESD technologies as well as the possibility of having them at multiple locations within the datacenter. Such placement should also take into consideration the volumetric/real-estate constraints that may restrict the usage/capacity of the ESDs at these locations (servers and racks) in the datacenter. Provisioning is closely tied with the associated control problem: how should these ESDs be used (i.e., charged/discharged) for a given datacenter configuration, utility tariffs, and power needs of various servers? Consequently, we design our framework to jointly address the provisioning and control problems. The goal is to determine an ESD based solution that maximizes the amortized net cost savings (i.e., cost savings in power-related cap-ex and op-ex minus cost of procuring and operating the ESDs).

Inputs

Workload (Power Demand): There is considerable prior work on characterizing and predicting server workloads (e.g., [7]) and properties such as time-of-day effects, etc., have been observed. For this work, we are concerned with a time-series of power draws at different levels of the datacenter. Since we already have a lot of ground to cover, we assume that prior work on load prediction can be leveraged, and combined with power modeling work (e.g., [17]), to derive a reasonable, accurate time-series of power draws at a server granularity over the given optimization horizon (e.g., a day). Further, our work is intended to provide guidelines when building (ESD solutions for) datacenters, at which point some estimate of load characteristics is in any case assumed for right-sizing of IT and power equipment. A more detailed treatment of these issues can be

considered in future work. Specifically, for server i, we assume its power demand time-series given by $P_{1,i,t}, t \in \{1, ..., T\}$, where $T \times \delta$ represents our optimization horizon. We will consider different such time-series - both synthetic and real workloads - in our evaluations.

Power Infrastructure (Cap-Ex): We assume L levels (e.g., datacenter, rack, server) in the power hierarchy and use the variable $l \in \{1, ..., L\}$ to denote a particular level, with $l = L$ corresponding to the highest level and $l = 1$ corresponding to the server-level. Within a level l, we denote the number of power supply equipment (e.g., transformers, switchgear, and centralized UPS at datacenter level, and so on until individual power supplies at the server level) by n_l. The equipment also includes any ESDs we may need to provision at that level, and we begin with homogeneous equipment at a level for simplicity, though this can be generalized (particularly when we have different parts of the datacenter running different workloads as in some of our experiments). Prior work has pointed to cap-ex costs ranging between \$10-20 per watt of power provisioning (i.e., for our set of $\sum_l n_l$ equipment). Though the costs are not explicitly stated for power provisioning at each level of the hierarchy, it is typically more effective to start under-provisioning from deep down the hierarchy (i.e., $l = 1$) since it would allow larger portion of the hierarchy to benefit from cap-ex savings. We consider a conservative cap-ex saving of \$10/watt resulting from peak shaving as a starting point, and go as low as \$1/watt in our experiments. This is the saving that would be obtained by reducing a watt from the maximum draw P_L^{max}. The power draw under l at any time is given by the sum of the power draws of all servers under this sub-hierarchy, and the maximum of this sum over our optimization horizon is denoted as P_l^{max}.

Utility Tariffs (Op-Ex): Utilities base their tariffs on the actual energy consumption (say a \$/kWh), and the need to sustain the maximum power draw across all their customers within the constraints of their existing capacity. To address the latter concern, utilities dis-incentivize high power draws (especially simultaneously from multiple customers) by two mechanisms: (i) vary a (say as $a(t)$) based on the time-of-day [18]; and/or (ii) track the peak draw (typically averaged over 15 minute windows) and impose a cost of b \$/watt (e.g., as in [30]). Our framework is generic enough to accommodate either, and we simply use mechanism (ii) in our discussions/evaluations. Consequently, we need to track P_L^{max} (i.e. the maximum power draw at any time at the datacenter scale), and associate a b \$/watt cost to it.

Optimization Problem Formulation

Decision Variables: Given our goal to jointly address provisioning and subsequent control, we choose decision variables that capture the operational aspects of ESDs as well as the decisions about their sizing and placement. In the subsequent discussions, the subscripts l and i of the variables indicate the level in the datacenter hierarchy, and the index of associated equipment instance at that level, respectively. E.g., at the leaf level, $l = 1$ and i can take values from 1 to the number of servers; at rack level, $l = 2$ and i can take values from 1 to the number of racks and so on. In general, the tuple (l,i) denotes the root of the sub-hierarchy governing a certain set of servers. A server can source its power only from the ESDs that are in the path from itself to the root. Subscript k is used to denote the ESD technology.

First, let $S_{k,l,i}$ denote the "size" - the energy capacity of an ESD of type k placed at (l,i). Second, for each such device, to capture its "usage", we use variables $D_{k,l,i,t}$ and $R_{k,l,i,t}$ to represent the discharge and re-charge rate, respectively, during time slot t. To carry over the residual ESD energy capacity from one time slot to the next, we use $E_{k,l,i,t}$, which is the energy left in this device at the beginning of time slot t. Finally, $P_{l,i}^{realize}$ denotes the realized peak as a result of our shaving in sub-hierarchy (l,i).

Objective: We can now express various components of our overall objective function. All of these have been normalized/amortized to the horizon of our time series. The expected cap-ex savings in power infrastructure due to under-provisioning is given as $CapExSavings = \sum_{l=1}^{L} \sum_{i=1}^{n_l} \alpha_{l,i} \times (P_{l,i}^{max} - P_{l,i}^{realize})$, where $\alpha_{l,i}$ is the savings for each watt of under-provisioning at level l. The expected op-ex savings can be expressed as $OpExSavings = (\sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{i=1}^{n_l} \sum_{t=1}^{T} a \times (D_{k,l,i,t} - \frac{R_{k,l,i,t}}{\eta_k})) + b \times (P_{L,1}^{max} - P_{L,1}^{realize})$, where a and b are the unit costs for energy and peak power draw in the utility tariff explained in previous subsection. Finally, the additional cost of ESDs themselves is given by $EStoreCost = \sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{i=1}^{n_l} (S_{k,l,i} \times C_{k,l,i})$. Here $C_{k,l,i}$ is the normalized cost of ESD k per unit energy adjusted to its actual lifetime, which depends on how the device is used (e.g., the same battery would last longer if it undergoes shallow discharges), and hence is itself unknown. Rather than dealing with a non-linear program in which $C_{k,l,i}$ is treated as an unknown, we keep our program linear and run it successively with the value of $C_{k,l,i}$ yielded by one run fed into the next run till convergence is achieved.

Finally, putting these components together, we have our objective as:
Maximize $(CapExSaving + OpExSaving - EStoreCost)$.

Constraints: We assume that all ESDs are fully charged at the beginning of the time-series, and need to leave them in the same state at the end of the time-series. In any time slot, an ESD may only hold energy between a lower threshold allowed by its recommended DoD and its maximum capacity. To capture these we have:

$$E_{k,l,i,1} = E_{k,l,i,T+1} = S_{k,l,i}, \forall k, l, i, (1a)$$

$$(1 - DoD_k^{max}) \times S_{k,l,i} \leq E_{k,l,i,t} \leq S_{k,l,i}, \forall k, l, i, t, (1b)$$

For each ESD, the amount of energy that can be discharged or stored is bounded by the product of its provisioned size and corresponding discharge ($r_k^{discharge}$) and re-charge ($r_k^{recharge}$) rates:

$$0 \leq D_{k,l,i,t} \leq S_{k,l,i} \times r_k^{discharge}, \forall k, l, i, t, (2a)$$

$$0 \leq R_{k,l,i,t} \leq S_{k,l,i} \times r_k^{recharge}, \forall k, l, i, t, (2b)$$

We account for the conversion losses (energy efficiency η_k) of an ESD, during the charging process as

$$P_{l,i,t} = \sum_{j=1}^{n_{l-1,i}} \left(P_{l-1,j,t} + \sum_{k=1}^{K} \frac{R_{k,l-1,j,t}}{\eta_k} - \sum_{k=1}^{K} D_{k,l-1,j,t} \right), \\ \forall l \geq 2, i, j, t, (3a)$$

Furthermore, when charging we should still ensure that the net power draw (including the power for all the equipment under l) is bound by $P_{l,i}^{realize}$ (which is in turn less than $P_{l,i}^{max}$). This gives us:

$$0 \leq P_{l,i,t} + \sum_{k=1}^{K} \frac{R_{k,l,i,t}}{\eta_k} - \sum_{k=1}^{K} D_{k,l,i,t} \leq P_{l,i}^{realize}, \forall l, i, t, (3b)$$

$$0 \leq P_{l,i}^{realize} \leq P_{l,i}^{max}, \forall l, i, (3c)$$

To account for energy losses due to self-discharge (μ_k) characteristics, we have

$$E_{k,l,i,t} = E_{k,l,i,t-1} + R_{k,l,i,t-1}\delta - D_{k,l,i,t-1}\delta - E_{k,l,i,t-1}\mu_k, \\ \forall k, l, i, t \geq 2, (4)$$

The ramp-up properties of ESDs pose restrictions on how fast the rate of discharge can itself increase over time (recall the analogy of car acceleration). We capture this as:

$$\frac{D_{k,l,i,t+1} - D_{k,l,i,t}}{\delta} \leq \frac{S_{k,l,i} \times r_{k,l,i,t}^{discharge}}{T_k^{ramp}}, \forall k, l, i, t, (5)$$

Our final constraints restricts the volume ($V_{l,i}^{max}$) within which the ESDs must be accommodated at various levels:

$$\sum_{k=1}^{K} \left(\frac{S_{k,l,i}}{v_k^{eng}} \right) \leq V_{l,i}^{max}, \forall l, i, (6a)$$

and

$$\sum_{k=1}^{K} \left(\frac{S_{k,l,i} \times r_k^{discharge}}{v_k^{pow}} \right) \leq V_{l,i}^{max}, \forall l, i, (6b)$$

Experiments and Evaluation

Configuration and Parameters: Our evaluations use a 4 MW datacenter of 8192 servers (placed in racks of 32 servers/rack), each with a 500W power supply, organized in a hierarchy as in Figure 4.1. We choose three levels for placing ESDs (L=3): top of the hierarchy (l=3, as in most datacenters today), rack-level (l=2, corresponding to some reported datacenters of Facebook [33] and Microsoft [79]), and server-level (l=1, as in some reported datacenters of Google [44]). Again note that these reported datacenters still place their ESDs at only one level, and not as a multi-level hierarchy which we allow.

Max. Power	4 MW
# racks	256
# servers	8192
V_2^{max}	20% of rack vol.
V_1^{max}	10% of server vol.
Cap-ex (α)	\$10,\$1/Watt
Op-ex energy (a)	\$0.05/kWh
Op-ex peak (b)	\$12/kW/Month

Table 3.2: Parameter Values.

We present results for $K = 5$ ESD technologies discussed earlier (however, LI based solutions do not explicitly appear in the results since whenever LI comes up on top of LA, the CAES option does even better). As discussed in Section 4.1.1, we use an op-ex cost model (Table 3.2) that is representative of that charged by Duke Electric [30], which has a monthly peak component charge of $12/kW/month in addition to the energy usage charge. We consider cap-ex cost of $10/W for the datacenter power infrastructure which is at the lower end of the $10-20/W range reported in literature [9, 50]. Note that even a cent of difference in additional cap-ex savings (i.e., $\alpha_{l+1} - \alpha_l > \epsilon$) when we push ESDs deeper down the hierarchy, will automatically enable our formulation to place ESDs deeper down the hierarchy as allowed by volume constraints. The cap-ex (12 year datacenter lifetime), op-ex (typically charged monthly), and ESD costs (appropriate actual lifetimes) are amortized for the horizon of the time series.

ESD Placement/Configurations: For comparison, we use 3 existing baseline strategies which are all "non-hybrid & single-level" placement styles (i.e., only one ESD technology x used at only one level in a given solution): $B_{dc,x}$, $B_{rack,x}$, $B_{server,x}$. Though these represent some datacenter ESD placement styles in use today, note that x=lead-acid batteries in most of them, and their capacities are chosen just for power outages. Our solutions will consider other options for x together with higher capacities. We compare these baselines with: (i) "hybrid & single-level," where we allow multiple technologies to be provisioned at a fixed level - centralized ($HybSin_{dc}$), rack-level ($HybSin_{rack}$), or server-level ($HybSin_{server}$), and (ii) "hybrid & multi-level," where we allow multiple technologies to be provisioned at possibly all of these levels ($HybMul$).

Workloads (Power Demands): To stress and understand the impact of workload parameters, we take the individual synthetic power demands P_1 ... P_{12} described earlier in section 3.1.2 and combine them in interesting ways (pair-wise) to bring out the impact of homogeneity and heterogeneity of workload behavior on ESD provisioning (section 3.1.3). Each resulting server-level power demand time-series spans a day, with each point in the series corresponding to the power needs over a one minute duration. Broadly, we refer to the peaks with mean width upto a minute as "narrow," 1-10 minutes as "medium," and higher (10 minutes to 2 hours) as "wide." In addition, Section 3.1.3 studies these issues with power demands of real-world datacenters and clusters (for Google [35], TCS [104], MSN [22] and stream media clusters [63]), reported in prior studies.

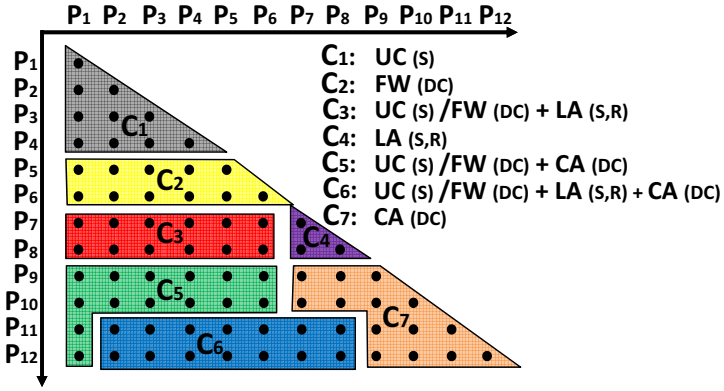


Figure 3.6: Results for the 78 synthetic workload combinations grouped in Clusters of "Best Configuration". The element (P_i, P_j) shows the ESD configuration when the per-server power demand is $P_{i,j}$.

Synthetic Workload Experiments While in Section 3.1.2 we had considered power demands that were homogeneous in time and in space (i.e., across servers) to get an overall idea of which single technology is better suited for a workload characteristic, we now use pair-wise combinations of the P_1 ... P_{12} time-series to study whether (i) hybrid combinations make sense, and (ii) whether multi-level ESDs are promising. This results in a 78-combination design-space of experiments to capture a large range of peak widths, heights and frequencies. This combination can create temporal heterogeneity. For example, Figure 3.7 shows one such power demand ($P_{2,9}$): it has two "phases" with significantly different properties. We will later see that this kind of workload is representative of some real world behavior with multi time-scale variations, e.g., time-of-day effects of load change at a macro-scale, with finer scale variations at each of these loads (as in the MSN workload). Dealing with such "temporal" heterogeneity, may mandate different solutions (hybrid or multi-level) compared to spatial heterogeneity, where different regions of the datacenter/racks may have different behaviors.

Results at a Glance: Figure 3.6 presents a simple way of viewing the results from our design space of experiments, by showing the best (or comparable to the best) ESD configurations for various $P_{i,j}$. For ease of discussion, we group these into "iso-configuration" clusters. The exact sizing decisions for various ESD technologies may vary across elements of a cluster. With a few exceptions, we find that our power demands fall into seven different clusters, and we present the ESD configuration for each cluster. For example, C_6 is best served by an ESD configuration which uses a combination of (i)

ultra-capacitors at server level or fly-wheel at datacenter, plus (ii) lead-acid batteries at server and/or rack levels, plus (iii) CAES at datacenter level. We find such a representation easier to parse, than go through every data point in the results. The savings from such ESD provisioning are typically between 10-40% of total datacenter power-related costs, after factoring in the cost of ESD provisioning itself. Below we discuss the provisioning choices made by our framework for each cluster, and give detailed cost savings results for the $P_{2,9}$ workload.

Temporal Homogeneity (Peak Widths are all Narrow or Medium or Wide): Recall that our analysis in Section 3.1.2 had suggested that ultra-capacitors are the most cost-effective in dealing with narrow peaks (P_1-P_4 in Figure 3.4) due to their low power cost, superior lifetime, and excellent charge/discharge ratio. With the additional volume constraints, self-discharge, and energy efficiency that our optimization framework captures, ultra-capacitors continue to serve as the most cost-effective ESD technology for such power demands, and suffice by themselves without requiring any other ESD technology. The capacity can be met right at the server-level, within its volume constraints (thereby providing higher cap-ex savings in the hierarchy). This set of results is depicted by cluster C_1 in Figure 3.6.

At the other end of the spectrum, we find that a solely CAES-based ESD design is best for demands whose peaks are wide (P_9-P_{12}), again in agreement with our model. However, the volume constraints mandate that its capacity be provisioned at the datacenter level, rather than rack/server levels. This set of results is depicted by cluster C_7.

However, in the middle of the spectrum, lead-acid battery at the server-level and/or rack-level is the most cost-effective ESD in dealing with less frequent medium width peaks (cluster C_4). However, when the frequency of such peaks increases, the lifetime deterioration of lead-acid batteries comes into play, making flywheel at the datacenter-level as the better option (cluster C_2).

Note that homogeneous peak widths of pair-wise collation of P_is refers to the region close to the diagonal in Figure 3.6. In these regions, creating power demand mixes is not significantly changing the individual demand properties, keeping these results in agreement with those in Section 3.1.2.

Temporal Heterogeneity (Mix of Narrow, Medium and Wide Peaks):

Clusters C_3, C_5 and C_6 bring out the need for hybrid, and possibly multi-level, ESD

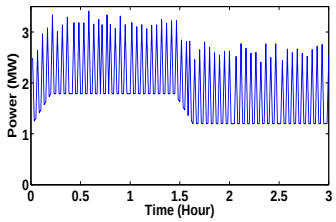


Figure 3.7: $P_{2,9}$ demand.

solutions. E.g., C_3 suggests both lead-acid batteries and ultra-capacitors at the server-level, or lead-acid batteries at server-level and flywheels at the datacenter-level; C_5 suggests CAES at the datacenter-level and ultra-capacitors at the server-level or both CAES and flywheel at the datacenter-level. In addition to the peak characteristics, the cap-ex savings when pushing ESDs deeper down the hierarchy as well as the associated volumetric constraints lead to these more extensive options rather than a single-level single-technology solution. To understand these issues in detail, we take $P_{2,9}$ (shown pictorially in Figure 3.7) and examine savings and ESD costs for different options in Table 3.3. We have the following observations and key take-aways from these results:

	B_ (savings, cost)	**HybSin_** (savings, cost)	**HybMul** (savings, cost)
Datacenter	LA (2.6k,0.5k)	UC+FW+CAES (3.0k, 0.2k)	FW+CAES
Rack	LA (2.3k,0.3k)	UC+LA (3.0k,0.2k)	-
Server	LA (1.7k,0.1k)	UC+LA (3.0k,0.2k)	UC (3k,0.2k)

Table 3.3: $P_{2,9}$: (Savings($/day), ESD costs($/day)). Total cost without ESD is $10K/day.

- Even the baselines $B_$ (i.e., current datacenters endowed with sufficient lead-acid battery capacities) can offer substantial cost savings (16-25% even after factoring in the storage cost). In this case, the savings are better with a centralized ESD that does not have volume constraints as opposed to restricted capacity sizes deeper in the hierarchy, since we do not have explicit cap-ex savings values when shaving peaks at each level of the power hierarchy. This lack of additional cap-ex information about

savings when we go deeper down the hierarchy (which is conservatively set to ϵ) is also the reason why the cost savings with $HybMul$ are not different from those of $HybSin_{dc}$ (though in reality the savings are likely to be higher for $HybMul$).

- For datacenters with centralized ESD, CAES (in combination with flywheels or ultra-capacitors if there are tall and narrow peaks to compensate for the slow ramp rate of CAES) appears to be a better option if space is not a problem. For instance, CAES-based hybrid datacenter level solutions provide 15% better savings than just a lead-acid centralized solution.
- The improvements that our `HybSin_` techniques offer over their corresponding baselines improve as we go down the hierarchy ($HybSin_{dc}$ offers 15% more cost savings that B_{dc} while $HybSin_{server}$ offers over 75% more cost savings than B_{server}). This results from the more stringent volume constraints at lower levels, where hybrid solutions that include ESDs with higher power and/or energy density offer larger gains. This may suggest that recent datacenters with distributed ESDs, like those at Google, Facebook and Microsoft, may benefit further from a move to hybrid ESDs.

Spatial Heterogeneity: Until now, we have only discussed spatially homogeneous power demands (i.e., all servers across the datacenter experience the same power demands). However, our techniques can address heterogeneous demands, and in fact hybrid and multi-level ESDs make even more sense in such environments. Many datacenters host different applications, with diverse power demands, e.g. a search engine may have applications which directly cater to Internet requests along with crawlers running in the background. These may not necessarily run on the same servers, and may even fall in different power sub-hierarchies in the datacenter. In the interest of space, we summarize one result to illustrate this point. When half the servers run P_1 and the other half run P_{12}, and we compare our $HybSin_{server}$ which uses ultra-capacitors and lead-acid batteries in the two sets of servers respectively, it provides over 50% more cost savings than both homogeneous ESD baselines $B_{server,UC}$ and $B_{server,LA}$.

Real Workload Experiments In this section, we construct power demands to mimic behavior reported previously in four different real-world datacenters/clusters.

TCS Figure 3.8(a) presents the power demand based on measurements reported from a TCS cluster [104] (that runs a range of many standard enterprise class applications)

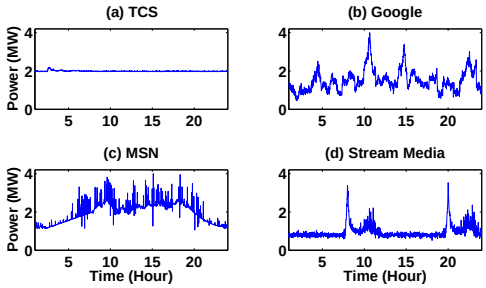


Figure 3.8: Real-world power profiles

scaled to our assumed 4 MW datacenter. This power demand has an extremely small peak (the peak to average ratio is about 1.1), which lasts roughly 2 hours each day. We find that the net savings offered by any ESD configuration is small (just 2%, though still off-setting the cost of the storage provisioning).

Google Figure 3.8(b) presents the power demand from a Google cluster [35] scaled to our assumed datacenter. This power demand shows several peaks during the day with high variances in the power demand, suggesting that ESDs can help. Furthermore, given that server-level LA ESDs are reportedly employed in their datacenters, we would like to examine whether we can improve upon them.

	B_ (savings, cost)	**HybSin_** (savings, cost)	**HybMul** (savings, cost)
Datacenter	CAES (4.9k,0.4k)	FW+LA+CAES (5.2k,0.3k)	FW+CAES
Rack	LA (4.7k,0.3k)	UC+LA (4.8k,0.3k)	-
Server	LA (3.9k,0.2k)	UC+LA (4.7k,0.5k)	LA (5.2k,0.3k)

Table 3.4: Google Workload: (Savings($/day), ESD costs($/day). Total cost without ESD is $12K/day.

Table 3.4 presents results for the ESD configurations under consideration. First, we see that if we are to use just a single technology, single-level solution, a datacenter level provisioning (using CAES) does provide considerably higher savings (nearly

25%) than using only lead-acid batteries at each server. The volume constraint does play an important role in limiting the benefits of ESDs in the lower levels of the hierarchy. However, considerations such as double-conversion, and more significant cap-ex benefits (than what we have conservatively used) across the layers, may be reasons for going with a server-level option as in Google. Our framework suggests that even if we are restricted to a server-level placement, a hybrid option of ultra-capacitors together with lead-acid batteries can mitigate the volume constraints to bridge this 25% cost savings gap. If we can remove restrictions even further and explore multi-level hybrid options, our framework suggests flywheel and CAES at the datacenter level, together with lead-acid batteries at the server level to provide as much as 30% benefits over just server-level batteries. As can be seen in Figure 3.8(b), this workload shows burstiness at different time scales, allowing a richer set of ESD options to shape this power profile.

MSN Figure 3.8(c) shows the load in a MSN facility [22], which has been translated to a power demand for our 4 MW datacenter. Table 3.5 shows the cost savings with the different ESD configurations. To a large extent, the results/savings are similar to those in the Google workload - capacity limitations and conservative cap-ex cost assumptions limit the extent of savings with just server-level lead-acid batteries, and CAES/flywheels can better augment the power needs centrally. The main difference is that this workload has very diverse sets of peaks - some last as long as 8 hours per day, and some which are at most a few minutes. The short and bursty tall power spikes favor a ultra-capacitor based solution at the server level, whether it be in $HybSin_{server}$ or $HybMul$. Overall we find $HybMul$ giving around 30% and 20% better savings than server level alone, or rack-level alone placement of lead-acid batteries.

Until now, we have presented only overall summary results. In order to give more detailed facets of ESD operation, we zoom in on a small time window of the MSN power profile and show the operation of different ESDs for $HybMul$ in Figure 3.9. For instance, the horizontal straight line shows the draw from the "utility", and we can see that despite the "demand" varying over time (which is also shown in this figure), the draw from the utility remains constant. To bridge this gap, the lines at the bottom show the charge (negative values of power in the y-axis), and discharge (positive values in y-axis) of the ESDs. We can see that CAES takes a bulk of the gap for significant portions of the time. However, when there is a sudden spike, the ultra-capacitor meets

	B_ (savings, cost)	**HybSin_** (savings, cost)	**HybMul** (savings, cost)
Datacenter	LA (4.0k,0.5k)	UC+FW+CAES (4.4k,0.3k)	FW+CAES
Rack	LA (3.8k,0.3k)	UC+LA (4.3k,0.3k)	-
Server	LA (3.4k,0.1k)	UC+LA (4.2k,0.2k)	UC (4.4k,0.3k)

Table 3.5: MSN Workload: (Savings($/day), ESD costs($/day). Total cost without ESD is $15K/day.

the difference. In order to charge this ultra-capacitor, we see that its curve goes through a negative spike just before it serves the required surge. Note that this negative spike is served by the energy sourced from the CAES, rather than pose an additional load on "utility".

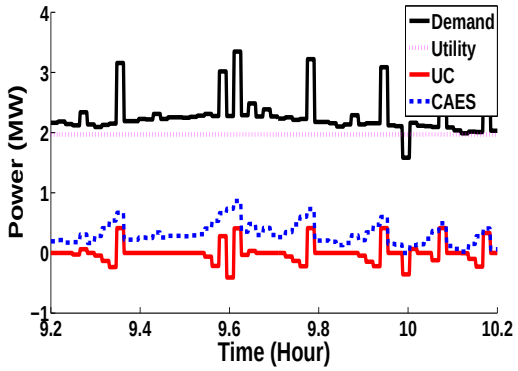


Figure 3.9: ESD charge/discharge control for the MSN power demand.

Next, we focus on $HydSin_{rack}$ to take a closer look at the impact of device lifetime and charge/discharge ratio on provisioning efficacy. As in the Google environment, we arrive at a hybrid solution that combines ultra-capacitors with lead-acid batteries. Of the three main ways in which ultra-capacitors are better than lead-acid batteries - lifetime, charge/discharge ratio, and power density - we would like to understand which precise ones allow ultra-capacitors to complement lead-acid batteries so well. To do this, we relaxed the constraints related to each of these one at a time, and compared those results with the solution in Table 3.5.

Streaming Media Temporal heterogeneity can also occur at shorter (than MSN) time-scales, as in a server running a streaming media server where there is a power spike once every hour or few hours when new clients login to start watching a new show. We see this in the Media server load [63] shown in Figure 3.8(d), which has again been scaled for our 4 MW datacenter. Table 3.6 shows the correspond savings with ESD options.

	B_ (savings, cost)	**HybSin_** (savings, cost)	**HybMul** (savings, cost)
Datacenter	LA (5.6k,0.2k)	FW+LA+CAES (5.7k, 0.2k)	FW+CAES
Rack	LA (5.6k,0.2k)	LA (5.6k,0.2k)	LA
Server	LA (4.0k,0.1k)	UC+LA (5.4k,0.1k)	LA (5.7k,0.2k)

Table 3.6: Streaming media: (Savings($/day), ESD costs($/day)). Total cost without ESD is $10K/day.

As before, if considering a single level and single technology provisioning, the savings are better at the higher levels because of the restrictions we have imposed. However, even the server level hybrid provisioning (comprising lead-acid batteries and ultra-capacitors) does as well as any centralized provisioning, since the peaks are not as wide as in the MSN workload - benefits of ultra-capacitors can out-weigh any volume constraints of lead-acid batteries. Adding higher level storage capacities does not buy much more (just around 6% improvement).

Key Insights from Evaluation

1. ESDs help reduce power-related cap-ex and op-ex by up to 50%, even accounting for their provisioning costs.

2. With a single-level restriction, real-estate constraints limit the savings that we can get with server/rack level distributed solutions. In such cases, centralized ESD placement is a better option (e.g. 25% better in the Google workload).

3. Allowing hybrid ESD technologies even at the server level, improves the savings by upto 35% (e.g. in MSN workload) compared to a single ESD option.

4. Overall, a multi-level multi-ESD solution provides the best savings, giving improvements between 10-30% with respect to the best single-level single-ESD solution.

Using a wide spectrum of synthetic workloads that stress different aspects of these ESDs and several real datacenter workload traces, we have shown (i) homogeneous ESD technologies suffice when there is not much heterogeneity in the workload, though the region of operation will decide which ESD should be deployed (e.g., narrow, tall and frequent peaks suited for ultra-capacitors/flywheels vs. broad and infrequent peaks better suited for compressed air and possibly batteries); (ii) even when placing ESDs in a single layer of the power hierarchy, considerations such as how much of the power hierarchy to optimize, volume constraints deciding storage capacity, and statistical multiplexing effects of the workload, influence where (server, rack or datacenter levels) the ESDs should be placed; (iii) even at a single layer of the power hierarchy, hybrid ESD solutions employing multiple technologies can offset the limitations of these constraints to provide substantial benefits (e.g., 25% improvement in cost savings in MSN at the server level); (iv) even if a hybrid ESD option is not employed at each level, a multi-layer hybrid solution can provide as much, if not better, savings across the spectrum of workloads that we have studied. The hybrid and multi-level ESD solutions are even more beneficial when the temporal (over different time scales) and spatial (across regions of the datacenter) heterogeneity in the workload increases.

3.2 System Support for Managing Energy Storage

In addition to the theoretical framework developed in the previous chapter, we also investigate research issues related to the design and implementation of systems software for managing energy storage devices. As power capacities - defined by cap-ex or op-ex constraints - get more and more stringent, treating power as a first class resource in datacenter management becomes essential. While there has been prior work [100, 82, 119, 10] on systems software for treating power on par with other resources such as CPUs and memory, such work has not really considered the energy storage dimension. primary focused on battery-based mobile devices [31, 65, 119, 120, 39, 57, 52, 19], where the goal has been to extend the operation time with batteries, unlike our goals stated here (peak, cap-ex and op-ex reduction).

We propose to systematically explore systems software mechanisms and policies for ESD-based solutions to managing datacenter power demand. We design and implement a software-based *virtual power hierarchy (vPower)* (from the datacenter at the root, to the servers) for each application, within which it can safely execute, insulated from any power-related emergencies arising from co-existing applications. The power hierarchy that is being virtualized includes the capacities of ESD of each layer and all the power equipments.

3.2.1 vPower System Architecture

Goals and Problem Statement: We work with an abstraction of the power hierarchy which has an imposed power cap P_i^{cap} at one or more levels L_i, where L_i= L_1 (server level) to L_L (datacenter level). Our focus here is only on mechanisms and policies to adhere to these imposed P_i^{cap} values, and a detailed evaluation of the trade-offs with different P_i^{cap}s (examined in prior work [46, 59, 48, 113]) is orthogonal to this work. Specifically, the implementation and evaluation platform in this work uses a 2-level hierarchy, L_1 (server) and L_2 (rack) with associated power caps P_1^{cap} and P_2^{cap}, but the discussions and results can generalize to more levels. At each level, we assume the presence of energy storage - our experimental platform uses lead-acid batteries, which are currently the most common in datacenter UPS units. The goal is to ensure that the power draw never exceeds P_i^{cap} at all corresponding L_is in the hierarchy, using a combination of (i) demand-response computing knobs - workload placement, consolidation, migration, scheduling, power state modulation (DVFS states), etc., and (ii) batteries with maximum power P_i^{bmax} and energy capacity E_i^{bmax} available at each level L_i in the hierarchy which can temporarily step in to provide the extra power needed to handle the needs beyond P_i^{cap}. Table 3.7 summarizes these notations. This work explores application interfaces, software mechanisms and policies to attain this goal when hosting multiple applications that share this power infrastructure. In the process, we need to adhere to application SLAs, meet these SLAs with the minimum amount of resources (i.e., maximize consolidation and utilization), and ensure fairness in how this power is allocated and shared between these applications.

Overview of System Architecture: Towards this goal, we implement a software-based *virtual power hierarchy (vPower)* (from the datacenter at the root, to the servers) for

Notation	Description
L_i	Power sub-hierarchy i
P_i^{cap}	Power cap of L_i
P_i^t	Power draw of L_i at time t
P_t^{req}	Requested power from an application for time t
P_i^{bmax}	Power capacity of a battery at L_i
E_i^{bmax}	Energy capacity of a battery at L_i

Table 3.7: Some Common Notations

each application, within which it can safely execute, insulated from any power-related emergencies arising from co-existing applications, even when their aggregated peak demand can exceed the power cap in any level. The power hierarchy that is being virtualized includes the capacities of all the power equipment (switchgear, transformers, UPS units, PDUs, and individual power outlets) from the root, all the way down to individual servers running this application. In addition, it also virtualizes the battery capacities of each layer.

While one could conservatively book/reserve for the peak power demands of applications, and admit only those whose collective peak demand fits within the power caps P_i^{cap}, overbooking can improve system utilization, since simultaneous and sustained peak demands from all applications may be less common. vPower has to: admit the right set of applications (*admission control* described in section 3.2.2); place/co-locate them with the right/synergistic mix of applications in the sub-hierarchy that would lead to fewer power emergencies (*placement* described in section 3.2.2); and suppress any power violations, if and when they occur, from propagating higher up the hierarchy using both demand-response computing knobs and the right batteries (*enforcer* described in section 3.2.2). An *accounting manager* needs to track the dynamic power consumption of different applications, as described in section 3.2.2, to provide up-to-date information for the enforcer to ensure fairness. As with any other shared resource management, vPower can leverage any application-level power related information that can be explicitly provided using a `palloc()` interface described in section 3.2.2.

The interfaces to the "power infrastructure hardware" that can be exploited by our system are shown in Table 3.8. Apart from interfaces to monitor the instantaneous power draw on any line in the hierarchy (`power()`), there are also interfaces to the batteries

in each level to monitor its state-of-charge (`soc()`), and control their charge and discharge rates. There are programmable power electronics circuitry to control the latter, but since our experimental setup does not provide this functionality, our evaluations use an empirically calibrated model of these rates of our experimental platform in the management decisions. In addition, the system also uses hardware/kernel/VM interfaces to change DVFS states, modulate scheduling and migration decisions, etc.

Interface	Description
`power(...)`	returns instantaneous power draw of given line
`soc(...)`	return state of charge of a battery
`charge(...)`	charge battery at specified rate
`discharge(...)`	discharge battery at specified rate

Table 3.8: Hardware Interfaces to Power Infrastructure

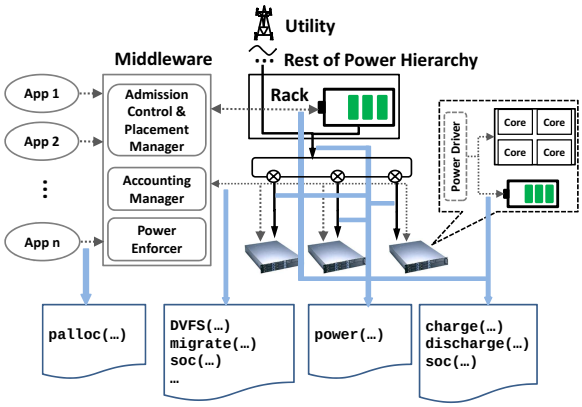


Figure 3.10: vPower Architecture

Our system architecture is pictorially depicted in Figure 3.10 for a 2-level hierarchy with server and rack level batteries, and power caps to be enforced at each of these two levels. There are two main software components: (i) one running as a driver within each server to initiate server level control knobs (DVFS, scheduling, etc.); and

(ii) another running as middleware on a dedicated server, which implements the management/control policies and appropriately interfaces with the server drivers. The latter also interfaces with the power distribution network to monitor (measure power draw, battery SoC, etc.) and control their operation.

3.2.2 vPower Mechanisms and Policies

The palloc() Interface As with any other resource, providing detailed and accurate information of an application's power profile can help manage it better, especially in aggressively under-provisioned settings. There are two dimensions to providing this information - how much? and over what duration? - and our application interface is accordingly specified as

```
palloc(<power,interval>,<power,interval>,...)
```

where the arguments are tuples specifying anticipated application power needs during different time intervals over its execution. Ideally one would like to accurate power needs at fine temporal resolution as is depicted in the power profile graphs of Figure 3.12. While many datacenter applications are long running services (e.g., web searching, memcached, etc.) and/or periodic/repetitive workloads (e.g., web crawling), with many of them going through profiling and fine-tuning phases before going into production mode [67], one may be able to obtain detailed and accurate power needs through profiling, barring execution and data dependency vagaries. Further, prior work on load predictability (time-of-day behavior in web services, flash crowd behavior for media services, etc.) may be useful for these specifications, as is prior work on phase characterization [51] to make such requests ahead of their need. However, getting accurate and fine-grained information (depicted as "Exact") may not always be feasible, and we accommodate several loose specifications (Table 3.9) both in requesting power, as well as in the temporal durations (possibly not even needing to specify durations) as below.

Along the temporal dimension, one need not even give any time information (as in PMax, PMin, PAvg, P90) if jobs are long running without much variance. It is also possible to estimate broadly defined execution phases (as in the MapReduce profile shown in Figure 3.12 where the power draws are quite different between map and reduce phases) and accordingly specify the intervals. In the interest of a uniform comparison across workloads, we introduce a range of pre-determined interval sizes between the "Every

	No Time	Entire Execn.	Every 60 secs.	Every Instant
Max. Power	PMax	PMax-entire	PMax-60	Exact
90th Per. Power	P90	P90-entire	P90-60	-
Avg. Power	PAvg	-	-	-
Min. Power	PMin	-	-	-

Table 3.9: Example palloc() Specifications considered in evaluations.

Instant" and "No Time" extremes, and consider representative intervals of entire application duration (PMax-entire, P90-entire) and 60 seconds (PMax-60, P90-60) in our evaluations.

Along the power dimension, one could consider a wide range of values starting from the peak (PMax) (which can be conservative depending on the time interval it is specified for), a 90th percentile (P90) of this peak (less conservative), an average over the interval (PAvg) or even as low as the minimum power (PMin). Note that our interface *does not* preclude an application from specifying any power value, over any particular duration. However, our system ensures that *regardless of what the application specifies (which is treated more as a hint), it will insulate applications from each other*. By giving bad hints - either intentionally or unintentionally - an application can at best hurt itself. For instance, when an application specifies values higher than PMax - this would come at the cost of our system possibly not admitting this application when power budgets are tight. At the other end, when an application requests very low power needs, and then (misbehaves) starts consuming higher power than what it initially specified, our enforcer will step in and penalize it.

The Middleware The middleware is the core software component for allocating and controlling power across applications. It consists of an admission control and placement manager, an accounting manager and a power enforcer. We next explain their goals, design tradeoffs and implementation details.

Admission Control and Placement Manager

Goal: Based on an application's palloc() hints, this manager evaluates whether it can accommodate these needs without violating existing allocations, and if so where in the

hierarchy it should be placed (i.e., co-location with others). The shared power infrastructure (even if the computing load is spread across different physical servers), and shared energy storage (batteries) at multiple layers in the hierarchy, introduces additional considerations to this problem compared to approaches that only consider computing resources.

Design Tradeoffs and Discussions: Note that, the net power draw of a hierarchy can be temporarily boosted beyond its provisioned capacity by as much as the *sum* of the power draws that can be sustained by all the batteries (distributed and hierarchical) within that hierarchy. Our admission control policies are therefore based on our reliance on these batteries to achieve power capping:

- *Conservative Policy:* In this policy, batteries are not taken into consideration for admitting applications. Batteries may still step in during execution to suppress peaks if they arise, which may be rare because of the conservative admission control. Hence, it may be better to use less stringent power needs in palloc() when using this policy.
- *Moderate Policy:* This allows a certain percentage (e.g., 20%) of battery power (P_i^{battAD}) and energy (E_i^{battAD}) capacity to be available in addition to normal line capacity when admitting applications. It can admit more, with possible subsequent emergencies when batteries run out.
- *Aggressive Policy:* This is the same as Moderate, with 100% (still leaves reserve capacity to ensure power availability mandates upon outages [47, 46]) of the capacity used for admission control. It is better to specify more stringent needs in palloc() (e.g., PMax or P90) in conjunction with this policy, to lessen emergency occurrences.

Once admitted, we need to decide where to locate/co-locate this application. With distributed batteries across the hierarchy, the degree of balancing/unbalancing their usage serves as the main criteria when exploring this question. This can be captured by the correlations - High, Low and Anti - of the power profile of this application with those in the sub-hierarchy/server where it is being considered. Co-location effectiveness is also a function of the admission control policy. Co-location of anti-correlated workloads may be a better option when conservative admission control is used, since the latter is less reliant on batteries, and the anti-correlation would lessen the possibility of power emergencies. At the other end, co-location of correlated workloads may be better for an aggressive admission control policy (which may have higher emergencies) since it aggregates (batches together) the discharging of batteries, giving longer time

windows for charging, similar to how unbalancing of load creates more opportunities for server shutdown in [88]. The design choices along these two dimensions are qualitatively summarized in Table 3.10 indicating the priority order of correlation degrees for placement under a given admission control policy. We will show experimental results to corroborate these choices.

Implementation: When an application specifies its power needs via palloc(P_t^{req},Δt), the admission control algorithm checks two aspects of this requirement - power (P_t^{req}) and energy ($P_t^{req} \times \Delta t$) needs - and is admitted only if both can be met. For applications which only specify power (e.g., PMax, P90, PAvg, PMin) due to the lack of time information, we deal with energy constraint on a best effort basis and simply assume the energy constraint is met for admission. These checks are recursively made starting from the root. For each sub-hierarchy L_i, a check is made to ensure: (i) for every time interval t, total allocated power ($P_{i,t}^{alloc}$) for existing applications plus the new power needs (P_t^{req}) from the requesting application should be less than or equal to the sum of provisioned power cap of this sub-hierarchy (P_i^{cap}) plus the amount of power from batteries (P_i^{battAD}) dedicated for admission control (as per the above 3 policies); (ii) total allocated energy for existing applications (E_i^{alloc}) plus the new application's energy needs ($\sum(P_t^{req} \times \Delta t)$) should be less than or equal to the sum of maximum energy from the outlet (E_i^{outlet}) and energy from batteries (E_i^{battAD}) committed to admission control. Different amounts of power/energy from batteries represent the aggressiveness of admission control policies as described above. Once admitted to a sub-hierarchy, we place an application based on the choices from Table 3.10.

	Conservative $P_i^{battAD} = 0$ $E_i^{battAD} = 0$	Moderate $P_i^{battAD} = 0.2P_i^{bmax}$ $E_i^{battAD} = 0.2E_i^{bmax}$	Aggressive $P_i^{battAD} = P_i^{bmax}$ $E_i^{battAD} = E_i^{bmax}$
High Corr.	(3)	(3)	(1)
Low Corr.	(2)	(2)	(2)
Anti-Corr.	(1)	(1)	(3)

Table 3.10: Placement choice for a given admission control policy. Priority order is (1), (2) and then (3).

Accounting Manager

Goal: It dynamically tracks an application's power with respect to its allocation across its servers.

Design Tradeoffs and Discussions: We adopt a simple credit based accounting mechanism [19] to track the difference between power reservation and its actual consumption for each application. One credit represents one watt of power. An application is given positive credits if it consumes (P_t) below its reservation (P_t^{req}) while credits are subtracted if its power goes above reservation. However, to prevent misbehaving applications from continuously banking credits without using them, a bound is enforced on the amount of credits that it can bank (e.g., the bound can be the battery capacity allocated to an application). Similarly, to prevent misbehaving applications from continuously discharging battery, a bound is enforced on the amount of credits that can be in debt.

Implementation: For each application, we maintain a data structure called PCB (Power Control Block), which contains its power needs, accumulated credits, assigned server(s), etc., and the credits are updated periodically. Power is sampled for each server (outlet of a Raritan PDU) at a second granularity via SNMP commands, which suffices if only one application is running on that server. Within a server, one could use apportioning techniques between co-existing applications using system metrics for power accounting as in [54, 100], though outlet level server metering suffices in our evaluations which places applications on distinct servers.

Power Enforcer

Goal: Despite admission control, there may be power emergencies during the runtime due to under-estimates of application needs and/or aggressive over-booking as explained earlier. The enforcer uses different computing demand-response knobs (DVFS/clock throttling and migration) and batteries in different layers of the hierarchy to handle these emergencies. Specifically, issues related to which knob(s) to use for different applications, which batteries to employ in a hierarchical setting, are some considerations in the design of the enforcer. The goal is to meet application performance SLAs, as well as maximize system utilization, while ensuring fairness between applications in the choice of knobs (performance detrimental computing knobs versus use of batteries).

Design Tradeoffs and Discussions: Consider any two hierarchy levels L_i and $L_{i-1,j}$,

with the latter having $j = 1..n$ components (and hence its 2-dimensional representation) directly connected to L_i. We use the following notations: the provisioned peak power (power cap) of the outlet that level L_i can draw from is P_i^{cap}; the required aggregate power draw by level L_i is $P_i^t = \sum_{j=1}^{n} P_{i-1,j}^t$ at time t; each component $L_{i-1,j}$ can draw power from its own battery with energy capacity $E_{i-1,j}^{bmax}$ and maximum discharge/charge power $P_{i-1,j}^{bmax}$, and some or all of the parent battery at L_i depending on the capacity of the provisioned line between $L_{i-1,j}$ and L_i. One or more of the $n+1$ (n at L_{i-1} and 1 at L_i) batteries can be used to shave all or part of the power violation at L_i.

Lemma. *No matter which strategy is taken to discharge batteries (i.e., which battery to use), a given peak P_i^t exceeding P_i^{cap} at L_i can be shaved by batteries in the hierarchy if the following three conditions are satisfied: (A) $(P_i^t - P_i^{cap}) \leq min(P_{i-1,j}^t, P_{i-1,j}^{bmax}), \forall j$, (B) a battery is discharged only when $P_i^t > P_i^{cap}$, and (C) discharging decisions of all batteries are coordinated (i.e., the aggregate power drawn from all $n+1$ batteries at instant t is at most $(P_i^t - P_i^{cap})$).*

Proof. We prove the Lemma by contradiction. Suppose that there exist two battery discharging strategies (S_1 and S_2) applied to the same hierarchy of batteries with the same power and energy capacity as well as state of charge for the same power and workload demand under the same power cap, lead to different peak shaving results: S_1 completely shave all the peaks while S_2 does not shave all the peaks. The reasons that strategy S_2 can not shave all the peaks are the following: (i) After draining up some batteries for prior peaks, the batteries left with energy do not have enough power capacity to shave the remaining peaks. However, this violates condition (A); (ii) Similarly, due to prior usage, only m batteries have residual energy and the workload assigned on these m servers are low so that even completely draw power from their batteries, the reductions are not enough to suppress the remaining peaks. However, this again violates condition (A); (iii) Due to prior battery usage, no energy is left in any battery while there are still remaining peaks. Since the strategy does not waste any energy given condition (B) and (C), this implies that the total energy available to S_2 is less than what is available to S1 which contradicts the assumption (both strategies applied to the set of batteries with the same state of charge). All these possibilities either contradict conditions or assumptions, and hence, prove the Lemma. □

This lemma implies that all batteries in the hierarchy can be treated as one large

battery placed at L_i provided the three conditions are obeyed. Condition (A) says that the violation at L_i can be shaved by removing the contribution from any child component $L_{i-1,j}$ by sourcing that component from its local battery. Hence, any one of the batteries at $i-1$ can be used to suppress the violation. Condition (B) says that the battery at $L_{i-1,j}$ is used only to address the violation at the parent L_i, and not to suppress a peak violation that happens at $L_{i-1,j}$ itself (i.e., only the parent is under-provisioned and not each child). These, together with perfect coordinated control of all child batteries to shave this peak (Condition C), gives the simplistic illusion of a centralized larger battery of energy capacity $E_i^{bmax} = \sum_{j=1}^{n} E_{i-1,j}^{bmax} + E_i^{bmax}$ and power capacity $P_i^{bmax} = \sum_{j=1}^{n} P_{i-1,j}^{bmax} + P_i^{bmax}$ at L_i.

However, in practice, these conditions may not hold. We need to, thus, carefully choose the right battery to handle the emergencies as described below to lessen the probability of violating these conditions. For clarity, we discuss these issues assuming L_i (parent) is a rack, and each child $L_{i-1,j}$ is an individual server.

Parent's violation is larger than power consumption of m children (i.e., $(P_i^t - P_i^{cap}) > \sum_{j=1}^{m} P_{i-1,j}^t$): If some policy schedules an imbalanced lower workload on m servers than the other $(n-m)$ servers, and has drained out the batteries of these $n-m$ other servers and the rack battery because of prior usage, then the power violation at the parent can not be handled by batteries alone. Even if these m servers completely draw power from their local batteries, the reduction will not suffice to address the emergency at the parent. The problem gets accentuated when m decreases, especially as we move towards the root. To address this concern, it is better for policies to (i) balance the load across servers to increase $P_{i-1,j}$ for under-utilized servers, (ii) use up the batteries at the servers which have less power demanding applications first before going to others in cases of load imbalance.

Parent's violation is larger than power suppliable by m children batteries (i.e., $(P_i^t - P_i^{cap}) > \sum_{j=1}^{m} P_{i-1,j}^{bmax}$): Consider a case where a policy leaves residual capacities in only these m batteries after prior usage, and has already drained out the other $n-m+1$ batteries (including the rack battery). In this case, the residual capacities of these m server batteries are not sufficient to handle the rack violation (again the problem accentuates when m decreases as we move to the root). To address this concern, it is better to (i) use batteries evenly at the servers, and (ii) save rack battery for such power violations since they are typically provisioned with larger power/energy capacity

to handle a potentially larger load.

Child is itself under-provisioned (i.e., $P^{t}_{i-1,j} > P^{cap}_{i-1,j}$ for some j): With aggressive under-provisioning deeper in the hierarchy, a child battery may have been used to shave its own peak, rather than the peak of a parent, thereby violating Condition B. Subsequently, when called upon to suppress the peak of a parent, there may not be residual capacity. Since a battery can be useful to suppress peaks from propagating higher in the hierarchy and not in the reverse direction, we use the following general guidelines in our policies: (i) reserve some capacity for suppressing emergencies at its own level, and (ii) use higher level batteries for the higher level violations as much as possible (provided lower levels stay within their budget).

Implementation: Note that palloc() from applications, in combination with our accounting manager, helps track the positive/negative credits accumulated by each application. Our runtime enforcer takes these credits into consideration when employing the computing and battery knobs in apportioning the emergency suppression mechanisms across applications. Such proportional power allocation (whether it be in the computing knobs or in the power from batteries), is similar to proportional fairness studied in other resources [108]. The enforcer prioritizes the order of employing knobs based on the impact on performance (e.g., migration can be relatively costly as studied in [48]), as well as the duration and stringency of the emergency.

When the estimated duration of violations exceeds a threshold, the power enforcer will start to migrate applications with least accumulated credits to destination hierarchies with sufficient slack. For smaller emergency durations, it uses local DVFS and battery knobs. Based on the guidelines of hierarchical battery usage discussed above, we use the hints coming from palloc() to implement the enforcer as follows: (i) reserve certain local battery capacity at each server if any local power violations are anticipated; (ii) use local batteries for smaller power violations, saving the shared higher level batteries for larger ones, and supplement it with DVFS if batteries alone cannot handle the need; and (iii) as far as possible, source from batteries of servers with low anticipated future demand.

When there is slack in power usage, batteries are re-charged in the following order: (i) batteries without sufficient charge on servers estimated to have local power violations are given first priority; (ii) batteries with lower state of charge are given higher priority; (iii) batteries on servers estimated to have low power demand (hence unlikely to incur

power violations) are given the least priority.

3.2.3 Evaluation of vPower

Experimental Setup

We evaluate vPower on a scaled-down prototype (Figure 3.11) using a cluster (rack) of $n = 8$ DELL PowerEdge servers with two Intel Xeon 3.4GHz processors each. The face-plate rating of these servers is 450W, idle power is around 120W and the peak power that we can push them to across our workloads is 300W. The dynamic power consumption can be modulated with 4 DVFS states (P-states: 3.4GHz, 3.2GHz, 3.0GHz, and 2.8GHz) and 8 clock throttling states (T-states: 12.5%, 25%, ..., 100%). The "Power Driver" at each server changes power states using the *IA32_PERF_CTL* and *IA32_CLOCK_MODULATION* MSR registers. Each server is directly connected to a 1000W APC UPS which, in turn, is connected to an outlet of a Raritan PDU. The PDU is then connected to another 4000W UPS serving as a rack-level battery, which is again connected to an outlet of another Raritan PDU.

Although we have a 1000W UPS unit connected to each server, for all our experiments we only assume a 300W UPS (close to the maximum power consumed by our server) and drain the UPS using a corresponding scaled-down runtime chart as in [46, 48]. We consider a 4-minute battery per server, of which we leave a residual capacity of 3 minutes for availability guarantees. The UPS reports its load, power draw and remaining battery runtime over an RS232 serial interface. The PDU can dynamically switch ON/OFF the supply to individual UPS units with SNMP commands over Ethernet, giving the illusion of a power outage to the downstream UPS to provide power from its batteries. This is a relatively conservative way of using batteries for peak shaving, than a more elaborate approach which has power electronics circuitry to instantaneously source only the additional power needs (excess over the cap) from batteries with the remaining coming from outlet power [46, 48]. vPower can leverage the server battery for enforcing server level caps, and can use one or more (and perhaps taking turns to extend the duration) server batteries and/or the rack battery for enforcing rack level caps. Our cluster has a shared NAS box which is mounted as a NFS storage volume by all servers. We use another cluster as the destination for migrating workloads. All our applications are hosted as VMs, with Red-Hat Linux 5.5, under Xen. A separate machine

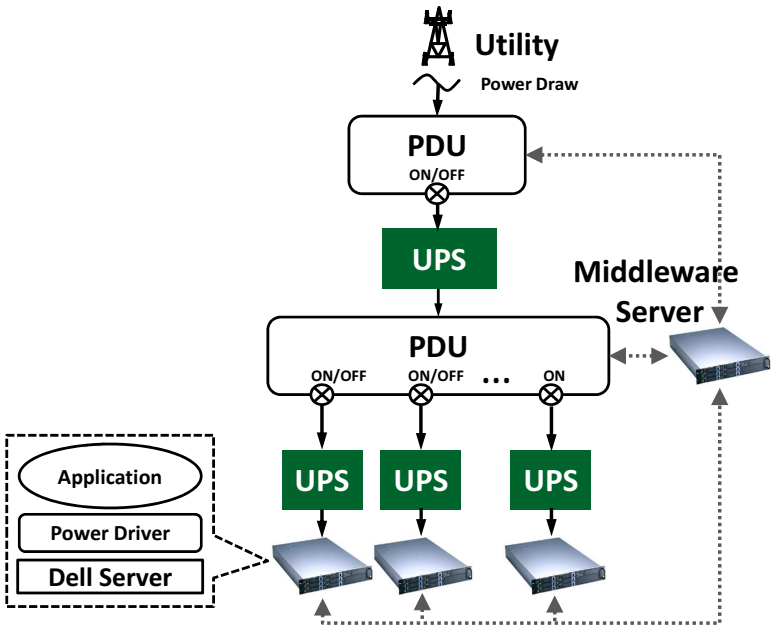


Figure 3.11: Experimental Setup

runs our "Middleware" which implements the algorithms, and sends appropriate throttling/migration commands to the server power driver and outlet turn on/off commands to the PDUs.

Workloads

We consider a suite of 8 representative datacenter applications (Table 3.11), that are both user-facing (interactive) and batch workloads. Interactive applications include the Yahoo! Cloud Serving Benchmark (YCSB) [25], an in-memory key-value store Memcached benchmark [76] and two other applications (MediaServer, WebSearch) from Cloudsuite [38]. Batch applications include a WebCrawling benchmark from Cloudsuite, a Hadoop MapReduce (word count used in several analytics applications), a GPU application in CUDA implementing the Black-Scholes financial model using a NVIDIA card, and a virus scanner (VirusScan). The last 2 are not amenable to migration from the server where they are executing. Figure 3.12 shows the power profiles on a representative server running these applications, and their runtime is given in Table 3.11.

Workload	Type	Runtime (Secs)
YCSB	user-facing	587
MediaServer	user-facing	370
Memcached	user-facing	1020
WebSearch	user-facing	629
MapReduce	batch	365
WebCrawling	batch	370
GPU	batch	1000
VirusScan	batch	888

Table 3.11: Workloads

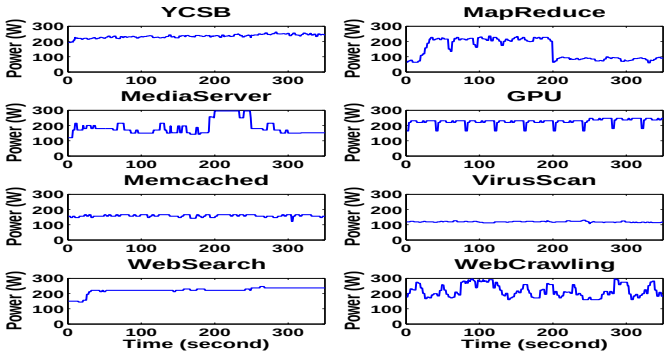


Figure 3.12: Application power profiles. For clarity, only first 350 seconds of execution is shown.

Evaluation Metrics

We consider the following metrics from the system and application perspectives:

- *Power violations:* the number of times that a power cap is violated at any of the two levels. Note that in our complete system, the enforcer will ensure no violations. We use this metric to compare the pros and cons of isolating the admission control/placement policies by removing the power enforcer in those experiments. In practice, the extent (magnitude and duration) of the violations should be tracked, but in the interest of clarity when presenting results we find that the number of power violations is a reasonable proxy for comparisons.

- *Degree of consolidation:* the number of applications that can be co-located on the same server/rack without exceeding the specified power caps. Consolidation can extract more value from existing compute and power infrastructure, and is also attractive from the viewpoint of lowering energy consumption/costs.
- *Scale-out:* the number of instances that an application can be replicated, to improve its throughput. This metric is more useful when the goal is to accelerate the performance of a single application (e.g., Memcached) as opposed to consolidation of disparate applications.
- *Performance Degradation:* the percentage degradation (of response time, throughput, completion time, etc. depending on the application) of running an application under a power cap with respect to the same experiment running without a power cap.
- *Fairness:* a measure of how the system treats co-existing applications from the performance viewpoint when meeting the power budgets. Rather than a single numerical metric, our results will clearly depict the differences between policies in penalizing applications.

Impact of Admission Control Policies

We begin by evaluating the three admission control policies - Conservative (II), Moderate (III) and Aggressive (IV) - and compare them with two baseline schemes - "Baseline-power" (I) which admits applications as long as adding their peak power consumption ($PMax$) does not violate the caps (i.e., assumes no statistical multiplexing or dynamic modulation knobs and is thus extremely conservative), and "Baseline-utilization" (V) which admits applications considering only the average utilization of the applications (which is somewhat representative of how consolidation is currently conducted, without regard to any power caps in the hierarchy, and is consequently very aggressive). As explained in section 3.2.2, the effectiveness of these policies depends on the specifications/hints that an application can give, and we consider the design space for the specifications given earlier in Table 3.9. In the following experiments, we set a rack level power cap of 1200W (approximately half of the maximum power to which we can push our rack). To isolate the impact of admission control, we remove the power enforcer in these experiments (which can lead to power violations depending on the aggressiveness of admission control), and study the resulting consolidation and scale-out effectiveness.

Consolidation: Admission control restricts the consolidation degree (the number of applications in a rack) with the possible benefit of fewer power emergencies. To study these trade-offs, we conduct a stress-test where we try to place as many of the eight applications (as allowed by the admission control policy) in the rack, and examine the resulting violations at the rack level. In reality, the choice for co-locating applications on a rack or a server would depend on additional issues such as performance interference, storage locality, security, etc., over and beyond power caps. However, we ignore such considerations to mainly isolate the impact of the power cap at the rack level, and allow up to 8 applications to co-exist in the same rack, but place each application on a separate server, i.e., there is no server level interference, but there could be rack level interferences.

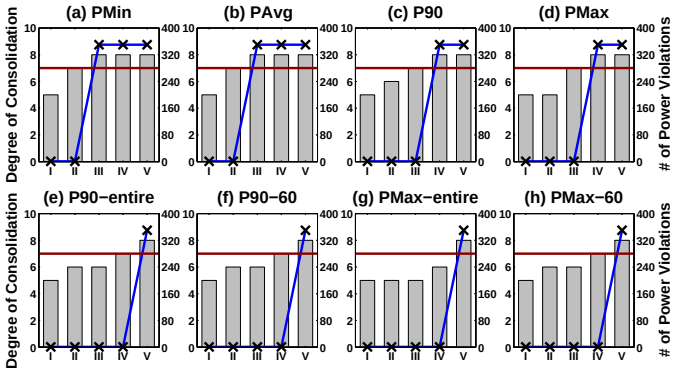


Figure 3.13: Consolidation Results for different Admission Control policies and Power Need specifications. Bars depict consolidation degree (left y-axis) and points on line depict number of power violations (right y-axis) of rack power cap set at 1200W. I:Baseline-power, II:Conservative, III:Moderate, IV:Aggressive, V:Baseline-utilization. Horizontal line represents consolidation degree (= 7) possible with "Exact" power specification without any power violations.

Figure 3.13 shows the trade-offs between consolidation degree and its consequences on power violations for each of the three admission control policies, comparing them with the two baseline extremes. Results are shown for the application power specifications outlined earlier in Table 3.9. In these graphs, we also show the maximum possible consolidation that can be attained (which is 7) without having any violations (and not requiring either batteries or computing knobs in the runtime), as a horizontal line. From these results, we make the following observations:

- A conservative admission control policy (II) which does not consider batteries, is comparable to "Baseline-power" (I) that provisions for the peak, and degenerates to the latter when the applications use the PMax specification (Fig 3.13 (d)). Unless applications grossly under-estimate their power (PMin in Figure 3.13 (a)), this policy is not preferable.
- At the other end, the aggressive policy (IV) achieves the same consolidation degree as "Baseline-utilization" (in the cases where a time interval is not specified).
- As long as the consolidation degree is less than or equal to 7, there are no violations, even when there is no dynamic enforcement i.e., statistical multiplexing of workload profiles suffices to keep the power draw within the cap. This is the case for all 3 policies in all the specifications which have a duration component (either "entire" or "60 seconds" in Figure 3.13 (e-h)).
- It is only when durations (even the execution time of the application) are not specified, that the moderate and aggressive policies start overbooking the power infrastructure, which can potentially lead to power violations - these would be suppressed at run-time by the enforcer with either battery or computing knobs. While this may appear counter-intuitive (i.e., no time component should lead to less flexibility in multiplexing needs and thereby lower consolidation), recall that there are 2 criteria to admission control - power and energy. Even though power multiplexing may be better with temporal information, note that when time durations are not specified, the energy criteria is always assumed to be met (section 3.2.2) from batteries, allowing more applications to be co-located.

Scale-Out: Instead of co-locating disparate applications within a given infrastructure capacity, a datacenter may only be interested in accelerating the performance of a single application with additional instances for scale-out. We conduct similar experiments by creating more instances of the Memcached server workload within the rack as allowed by the admission control policies. Figure 3.14 (a) shows a representative result for the P90 specification, which reiterates the observations made in the consolidation studies. With more aggressive admission control, we can add more instances, with the number of memcached servers supported by the rack going to 8 for the Aggressive (IV) policy. Violations also increase, but we will shortly show that these can be effectively managed without significant performance degradation when we introduce the enforcer. Without

any runtime enforcement, one can achieve a scale-out to at most 5 servers (horizontal line) if we are constrained by the power cap.

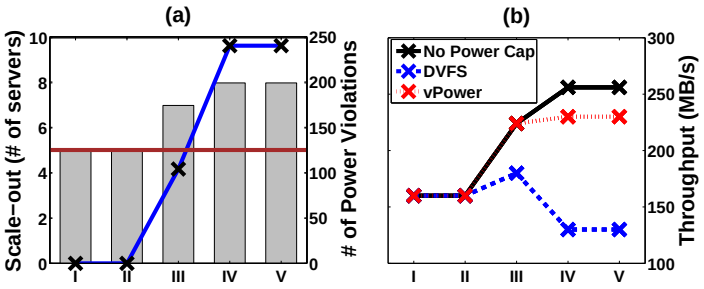


Figure 3.14: Memcached Scaleout with P90. I:Baseline-power, II:Conservative, III:Moderate, IV:Aggressive, V:Baseline-utilization. In (a), the horizontal line represents scale-out degree (= 5) using "Exact" power specification without any power violations, bars depict scale-out degree, and points on line represent number of power violations.

Impact of Placement Policies

We conduct experiments to study the impact of the policies when placing two applications - WebCrawling and MediaServer - in the same rack, with each application placed on its own server. These two are representative of more sinusoidal behavior in the power profile, helping us to study the influence of cross-correlations. To capture different cross-correlations, we simply vary the starting time of these applications, helping us capture a wide range of multiplexing possibilities (correlations). The "P90" interface is used to specify the power needs of these applications, and the enforcer is in place to avoid violations at the potential cost of performance degradation. As explained in section 3.2.2, placement choices go hand-in-hand with admission control, and we consider the correlation factor interactions with the aggressiveness of admission control as is depicted in Table 3.12. *Note that the power caps need to be changed in order to ensure the two applications are admitted in all the admission control schemes, i.e., power cap is increased by the amount of battery capacity as you move from aggressive (right column where the power cap is 300W) to conservative (left column where the power cap is 500W) in Table 3.12. Consequently, one should not compare the performance degradations across columns.* Rather, the trend within each column, and how that trend changes when we move from aggressive to conservative is what is important. As we can

see, the columns to the left prefer an anti-correlated co-location of workloads, while the aggressive admission control prefers a more correlated placement. These observations validate our choice of placement priorities based on the admission control policies as discussed earlier in section 3.2.2.

	Conservative	Moderate	Aggressive
Rack Cap	500W	460W	300W
High corr.	(1,1)	(8,6)	(15,14)
Low corr.	(0,0)	(0,0)	(20,18)
Anti-corr.	(0,0)	(0,0)	(25,20)

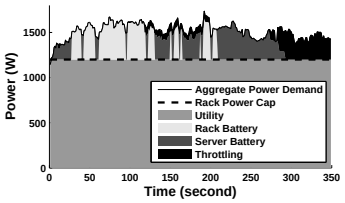
Table 3.12: Performance degradation of placing WebCrawling + MediaServer in the same rack with different cross-correlations and admission control policies. Tuples (WebCrawling, MediaServer) show % degradation w.r.t. running the same experiment without a power cap.

Effectiveness of Enforcer

Performance Results in Figures 3.13 and 3.14 (a), showed the impact of the admission control policies without the enforcer in place, which can lead to power violations in some of the cases. We now reinstate the enforcer to suppress these violations, and show the resulting application degradation for those two sets of experiments with the "P90" specification in Figure 3.15 (a) and Figure 3.14 (b) respectively.

Application	DVFS	vPower
YCSB	49	20
MediaServer	24	9
Memcached	49	25
WebSearch	107	48
MapReduce	72	41
GPU	60	25
WebCrawling	39	10
VirusScan	0	0

(a) Performance Degradation (%)



(b) Power Profile (Sourcing and Capping)

Figure 3.15: P90 specification in the Figure 3.13 experiment with Enforcer

As can be seen, while the degradation is non-zero in most applications for the consolidation experiment (Figure 3.15 (a)), it is still significantly better than a DVFS-only approach (an Oracle-based best DVFS states are chosen to adhere to the power caps). Degradation with vPower is between 15% to 69% lower than in a DVFS-only approach

for the more power hungry applications (VirusScan draws relatively low power). The effectiveness of the enforcer can be explained with Figure 3.15 (b), which shows how it meets the aggregate power demand of all 8 applications - through normal power (utility), one or more batteries, and DVFS throttling when necessary. With vPower, DVFS is employed mainly when batteries do not suffice (from 270 seconds onwards). There are periods in-between (e.g. 120-200 seconds), when some applications reach their credit limits, and DVFS is used to enforce their debt, as seen in a small top portion of the curve in this region. Server batteries are the first choice for small amplitude violations while the rack battery is used for most large amplitude violations. The choice of which battery to use, and the fairness to differently meet application demands (through batteries or throttling) is explained in more detail later in this section. In the shown zoomed-in 350 second time window, there is little opportunity for re-charging the batteries, though such re-charging does happen with power slack from 370 seconds (not shown in Figure 3.15 (b)).

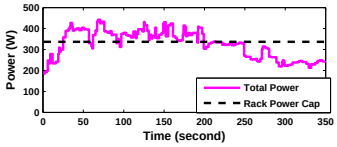
Similarly, the scale-out experiment for Memcached in Figure 3.14 (b) shows vPower giving throughput close to the uncapped case, and a value that is 25%-75% higher than the throughput of a DVFS-only option. Note that for aggressive policies (IV and V), the throughput of DVFS-only enforcement actually drops below the throughput of less aggressive policies (I, II and III); while the throughput with vPower actually increases despite the power cap. This shows that vPower can help applications such as Memcached achieve better scaleout capabilities when hosted in aggressively under-provisioned power infrastructure.

Battery Management We now show two examples depicting the importance of picking the right batteries at the right time towards maintaining the power caps, and show how the enforcer in vPower chooses the better option than always opting to first use the rack battery or server batteries.

Rack (Shared) Battery First being the better option: We run an experiment with a rack of two servers running MapReduce and MediaServer respectively, and the aggregate rack level power draw is shown in Figure 3.16 (a). Both applications specify their power needs using “P90-60”, and we set both a server-level power cap of 240W and a rack level power cap of 340W (we proportionally reduce the amount of power that can be drawn from the rack level battery, which was originally provisioned for 8 servers).

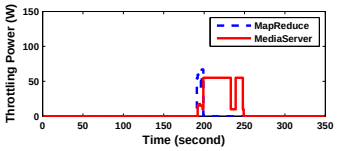
As per heuristics described earlier in section 3.2.2, when the enforcer anticipates local power violations, it first discharges rack (shared) battery for rack power violations and uses server level (local) batteries for such violations only when the shared battery reaches its lower threshold. This leaves more charge in local batteries for later use, and Figures 3.16 (e) and (f) show the amount of power that is capped for each application by throttling and the state of charge (SoC) of batteries (at the two servers and at the rack), respectively with vPower. On the other hand, a scheme (local battery first) that discharges local batteries greedily and goes to the rack battery only when the former runs out of charge, will not be able to handle the higher load that may come later to exceed the server level power cap. This can be seen in the throttling power and SoC graphs in Figures 3.16 (c) and (d) respectively. Using up the MediaServer server's battery in the first 200 seconds to handle rack level violations, leads to its inability to handle its own power violations later on, resulting in a 9% performance penalty (Figure 3.16 (b)). Note that we intentionally leave a residual capacity of 20% of the usable battery capacity (which already excludes residual capacity for availability guarantees) due to battery lifetime issues [113]. Interestingly, MapReduce is throttled by vPower at around time 200s, despite having capacity in its local battery. This is because it is in debt, and its local battery is being conserved for possible subsequent rack level violations, in the interest of fairness that is covered later in this section.

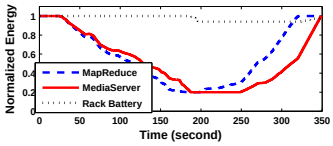
Server (Local) Battery First being the better option: On the other hand, Figure 3.17 (a) shows the aggregate power of three applications (Memcached, GPU and VirusScan) running on a server each in the rack, specifying their power needs using "P90-60". We intentionally reduce the runtime of VirusScan to 200s and show results for a duration of 450s. Only a rack level power cap is set at 450W and we proportionally reduce the amount of power that can be drawn from the rack battery as in the previous experiment. vPower anticipates no local power violations, and also anticipates an earlier completion of VirusScan with the P90-60 specification. Based on its heuristics, it uses local batteries first to handle rack level power violations in this case, and in fact prioritizes the draw from VirusScan server's battery before it finishes (Figure 3.17 (f)), removing any necessity for throttling (Figure 3.17 (e)). However, obliviously using a shared battery first approach in this case mandates subsequently throttling Memcached and GPU, causing performance degradation of 15% and 8% in these applications, respectively.



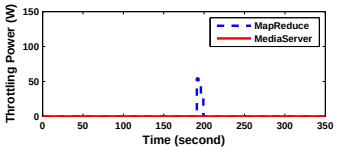
	Local-First	vPower
MediaServer	9	0
MapReduce	1	1

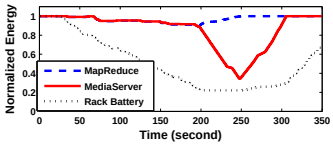
(a) Aggregate Rack Power (b) Performance Degradation (%)





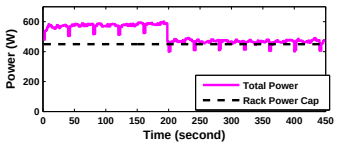
(c) Throttling (Local First) (d) SoC (Local First)





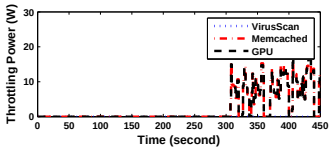
(e) Throttling (vPower) (f) SoC (vPower)

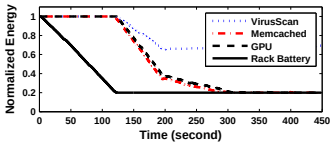
Figure 3.16: Shared Battery First is Better Option (MediaServer+MapReduce)



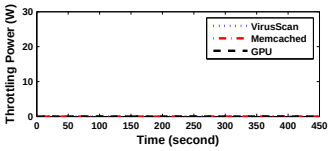
	Shared-First	vPower
VirusScan	0	0
GPU	8	0
Memcached	15	0

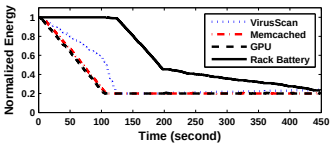
(a) Aggregate Rack Power (b) Performance Degradation (%)





(c) Throttling (Shared First) (d) SoC (Shared First)



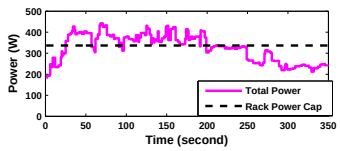


(e) Throttling (vPower) (f) SoC (vPower)

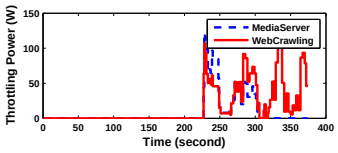
Figure 3.17: Local Battery First is better Option (Memcached+GPU+VirusScan)

Fairness We finally evaluate vPower's ability to enforce isolation between power draws of different applications for fairness. As noted earlier, there are pros and cons in the stringency of power demands made by an application. Asking for very stringent power may lessen their chances of being admitted, making them wait longer for power allocation. At the other end, even though grossly under-specifying the power may get them admitted, vPower will ensure that such applications do not mis-behave/misappropriate much more power in the runtime at the expense of others. The accounting mechanism in vPower, enforces a bound on the amount of credits that can be banked, as well as a bound on the credits that can be in debt. To illustrate these issues, we conduct an experiment with WebCrawling and MediaServer, each running on its own server. WebCrawling uses PMin=180W for its power specification (a significant understatement), while MediaServer uses PAvg=180W. Rack power cap is set to 325W, with no individual server power caps. We show the benefits of vPower's accounting mechanism by comparing its execution with that for a scheme which uses DVFS and batteries as vPower, but without the accounting mechanisms as shown in Figure 3.18. Without accounting in place, power needs of WebCrawling are continuing to be met (there is no throttling for it for the first 225 seconds) through batteries, and both applications are being penalized subsequently (12-13% performance degradation). MediaServer is being penalized in this case for WebCrawling's fault, making it unfair. On the other hand, with our accounting mechanism in place, WebCrawling's credits deplete rapidly, while MediaServer saves/accumulates its credits. Consequently the penalization for WebCrawling (which is mis-behaving) steps in a lot sooner, and MediaServer is not affected at all.

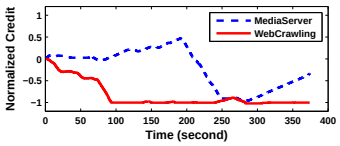
In summary, vPower allows applications (not essential) to explicitly request their power needs at different resolutions using a palloc() interface. We have shown that such information can considerably help system performance, while still shielding individual applications from explicitly managing this resource, similar to the malloc() analogy. vPower uses these specifications in determining whether to admit them, and if so, where to place them. We have shown that ignoring the battery capacities, and conservatively allocating for the potential peaks to avoid any power emergencies, results in over 30% reduction in system utilization. This also highly limits the scale-out capabilities of applications such as memcached. Consequently, an aggressive admission control policy that places correlated workloads together - to offer more opportunities for batteries to recharge - is a better policy. Despite an aggressive admission control policy, vPower's



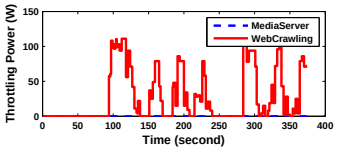
(a) Aggregate Power



(b) Throttling (vPower w/o Account.)



(c) vPower Account.



(d) Throttling (vPower w/ Account.)

Application Name	w/ Account.	w/o Account.
MediaServer	0	12
WebCrawling	28	13

(e) Performance Degradation (%)

Figure 3.18: Insulation from Power-hungry Applications (MediaServer+WebCrawling)

enforcer is able to effectively manage the power violations. We have shown that in our experimental rack of 8 servers, vPower shows 15-69% lower performance degradations than for a scheme that uses purely demand-side computing knobs, on a power infrastructure which is 50% under-provisioned. It also provides at least 50% higher throughput than the latter, for a memcached scale-out workload in this aggressively under-provisioned system. We have also demonstrated that the choice of batteries to draw upon during the runtime in a hierarchical 2-layer setting is very important. Based on a theoretical framework that identifies the conditions when this decision making becomes important, we have incorporated heuristics into vPower for battery sourcing. We have shown that vPower does much better than greedily using up local batteries or shared batteries first. Finally, we have demonstrated the effectiveness of vPower in fairly treating applications, by insulating the (intentional or unintentional) misbehavior of one application from the power needs of another. Our contributions are applicable regardless of where batteries are placed (centralized, distributed or hierarchical at multiple levels), the choice of energy storage technologies, and battery capacities. In fact, more stringent battery capacities make it even more important to manage hierarchies intelligently.

Chapter 4

Energy Storage for Underprovisioning Backup Power Infrastructure

In the previous chapter, we have explored re-purposing existing energy storage in the form of UPS battery units for demand response. Especially, focusing 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 chapter, we focus on the primary usage of energy storage in today's datacenters for handling power outages, investigate the relatively unexplored area of underprovisioning, and perhaps even removing, parts of the backup power infrastructure such as diesel generators.

4.1 Design Space of Backup Power Infrastructure

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 dissertation, 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.

4.1.1 Backup Infrastructure Cost Analysis

Power Hierarchy and Backup Infrastructure: Power enters the datacenter from the utility substation (Figure 4.1). 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 work).

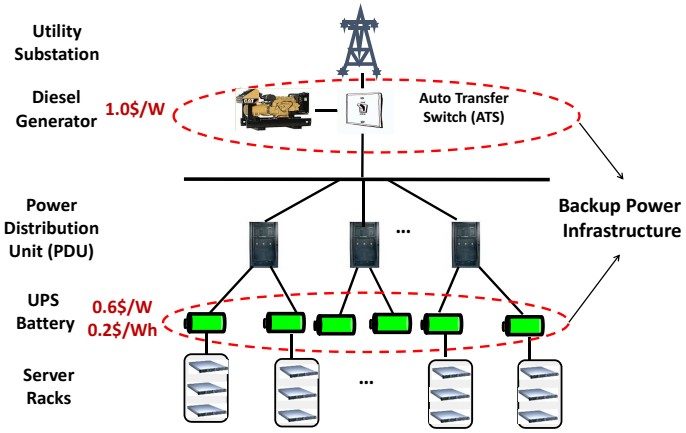


Figure 4.1: Datacenter Power Infrastructure

Figure 4.1 shows UPS units placed at the rack-level which is popular in today's datacenters (as in Facebook [34] and Microsoft [78]) 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 parallel), where the latter is preferred in today's datacenters to avoid double-conversion inefficiencies [64] associated with online UPSes. Unlike *online* UPSes which seamlessly transfer to sourcing from their batteries upon a utility

[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 work.

[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 work. One could possibly leverage mobile substations [11] with rapid on-demand deployment (with 12 to 24 hour advance notice) for maintenance purposes in case of non-/under-provisioned DGs.

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 [81]. 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 start-up 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 [6]. 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 1.2).

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 [113]), 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 4.2. 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

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

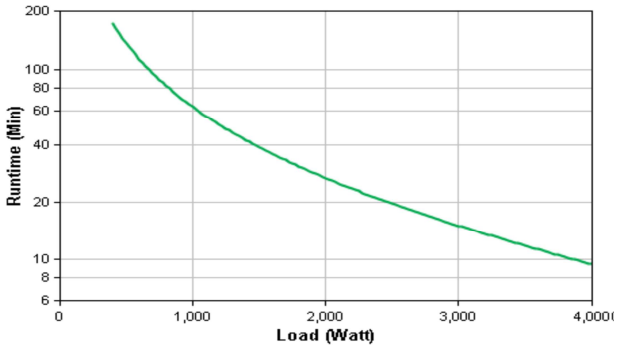


Figure 4.2: Runtime for a battery with max. power of 4KW.

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:

$$DGCost = DGPowerCost \times DGPowerCapacity \tag{4.1}$$

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:

$$\begin{aligned} UPScost = UPSPowerCost \times UPSPowerCapacity + \\ UPSEnergyCost \times (UPSEnergyCapacity \\ - (UPSPowerCapacity \times FreeRunTime)) \end{aligned} \tag{4.2}$$

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 4.1 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 [5]. The battery *FreeRunTime* assumes the rack-level battery placement and rated power capacity to supply a rack (in multiple of kilo-watts). 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 [9].

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

Table 4.1: DG and UPS cost estimation parameters [5, 9]. 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 4.2: Estimated amortized cap-ex annual cost of backup infrastructure for different datacenter capacities. M$ indicates million dollars.

Table 4.2 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.

4.1.2 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 4.3: 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 4.3 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 4.1.1). 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 4.3, 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 4.3 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 *reduced 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 4.3) 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.

4.2 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 4.3, and discuss them below.

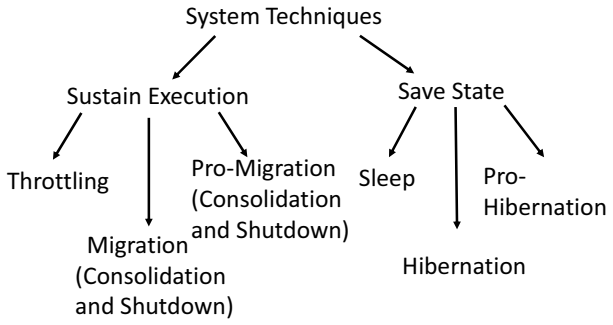


Figure 4.3: Techniques for realizing different application performability goals during power outages.

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 4.4: Performance and Availability implications of underprovisioning techniques.

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 4.5: 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).

Throttling: The application may be operated in a lower performance mode using active CPU power states (P/T states) [115, 41, 62, 91, 27]. 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 [20, 88, 61, 45, 118, 100]. 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

[4] Recall from section 4.1.1 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.

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 [24].

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 [98] 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 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 [26] 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 [95, 73, 72, 74, 1, 117]. 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 work, 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 4.4: (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 4.4 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 4.5. 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 4.2). Among the basic techniques identified above, *Throttling* is the only technique that is guaranteed to reduce

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 4.6: Hybrid Sustain-Execution + Save-State techniques.

the peak power (refer Table 4.5) (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 4.6 to refer to these hybrid combinations. The consequent 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 4.6 in our evaluations.

4.3 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 [24] and Remus implementations [26] 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 [24]) 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 4.7) that have different kinds of state, and consequently demands on the backup infrastructure for availability:

- Specjbb [99] 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 [75] 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 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 4.7: 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) cap-ex based on the cost model in Section 4.1.1. We normalize the cost of the different under-provisioned configurations to that of today's datacenter configuration (*MaxPerf* in Table 4.3).
- *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 4.7 lists the application specific performance metrics. For each application, we normalize its performance to that in *Max-*

Perf. 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.

4.3.1 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 4.3. For each backup configuration, we choose the system technique (from Tables 4.4 and 4.6) that offers the highest performance and lowest down time.

Figure 4.4 presents the 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 4.3), 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 4.4(b). *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*.

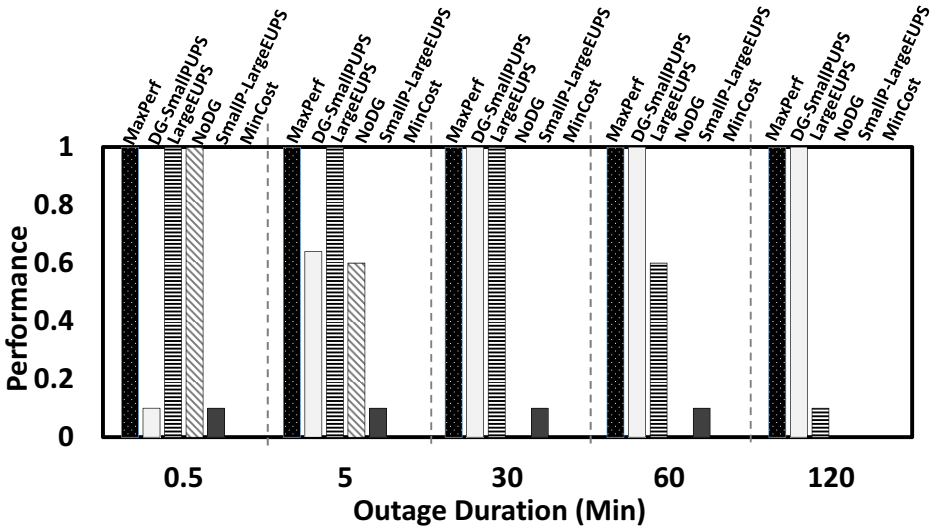
SmallDG-SmallPUPS (not shown for clarity) in Table 4.3 also has zero down time but at higher performance penalty. Configurations without DG cannot avoid down time during long outages.

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 4.4. 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).

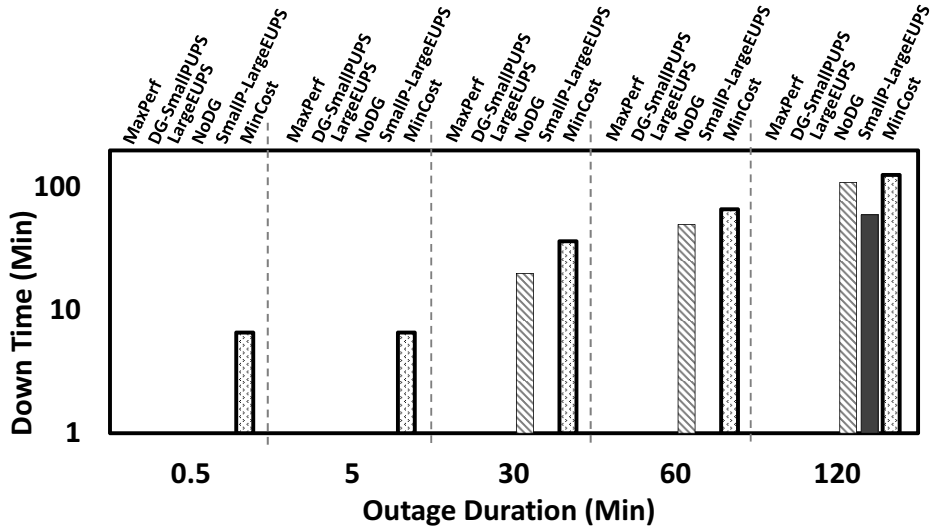
Summary of Insights: (i) Though DG translates long outages into small ones from the perspective of offered performability, it does so at a significant cost. (ii) UPS plays a crucial role in improving performability for short outages irrespective of the presence of DG. (iii) UPS can eliminate DG for up 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.

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 4.8: 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.



(a) Performance of configurations under different outage durations.



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

Figure 4.4: Cost and performability tradeoffs between the 6 configurations - *MaxPerf*, *DG-SmallPUPS*, *LargeEUPS*, *NoDG*, *SmallP-LargeEUPS* and *MinCost* - for Specjbb.

4.3.2 Effectiveness of Outage Handling Techniques

We now compare the performability-cost tradeoffs offered by different system techniques listed in Tables 4.4 and 4.6. In the interest of clarity, we first show representative results for Specjbb, and then show specific results for the other applications. Recall from Section 4.3.1 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 4.5 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 [8]. 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 [48]. *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).

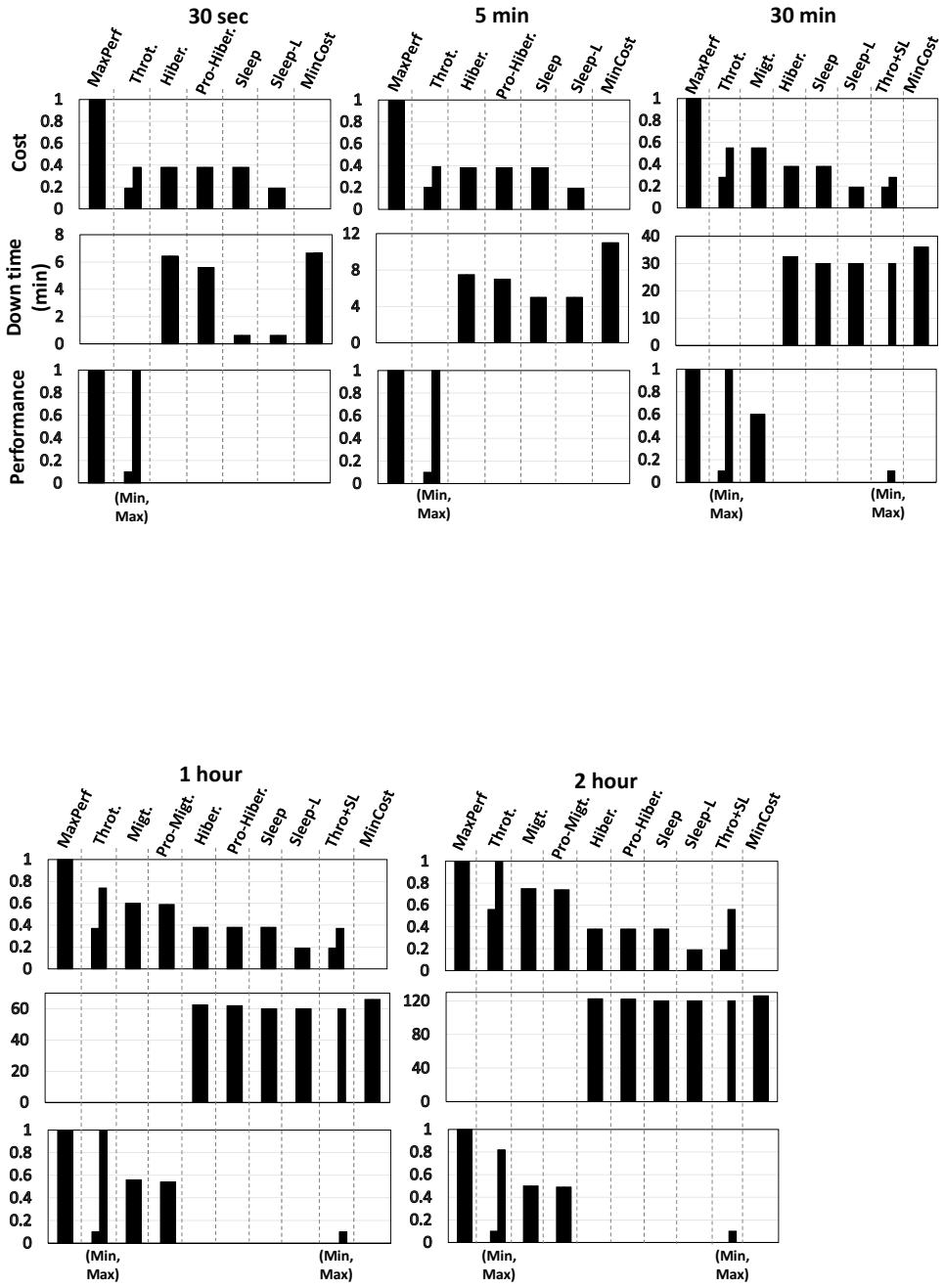


Figure 4.5: Power outage duration impact on different techniques for Specjbb. In techniques which use DVFS Throttling, the minimum and maximum values are shown to illustrate the range based on the chosen voltage-frequency states.

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 4.8 shows the measured save and resume duration 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 4.5. *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 4.1.1). 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, 53].

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. 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 4.6): 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 [115]. *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.

Web-search (Figure 4.7): 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

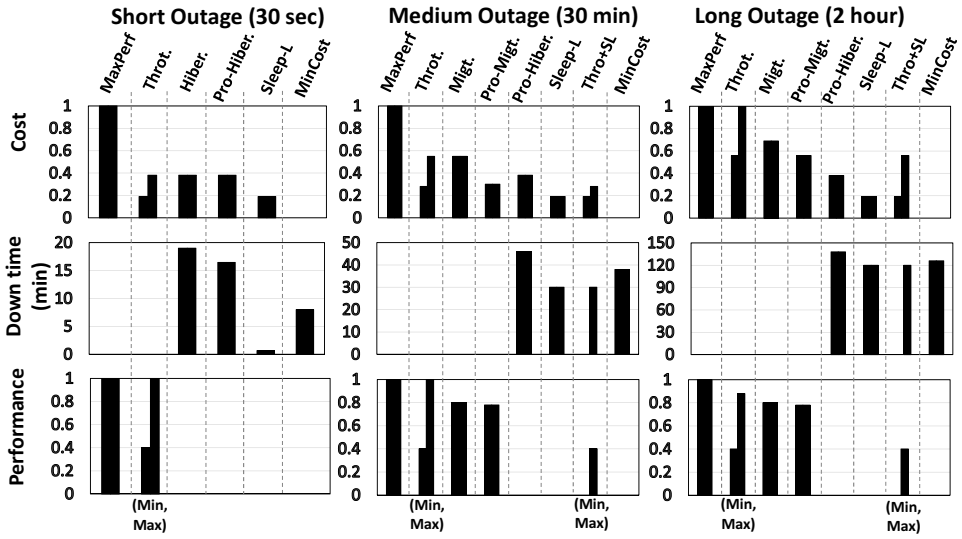


Figure 4.6: Tradeoffs for Memcached

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 4.7. 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 4.8): These scientific applications may run for several hours be-

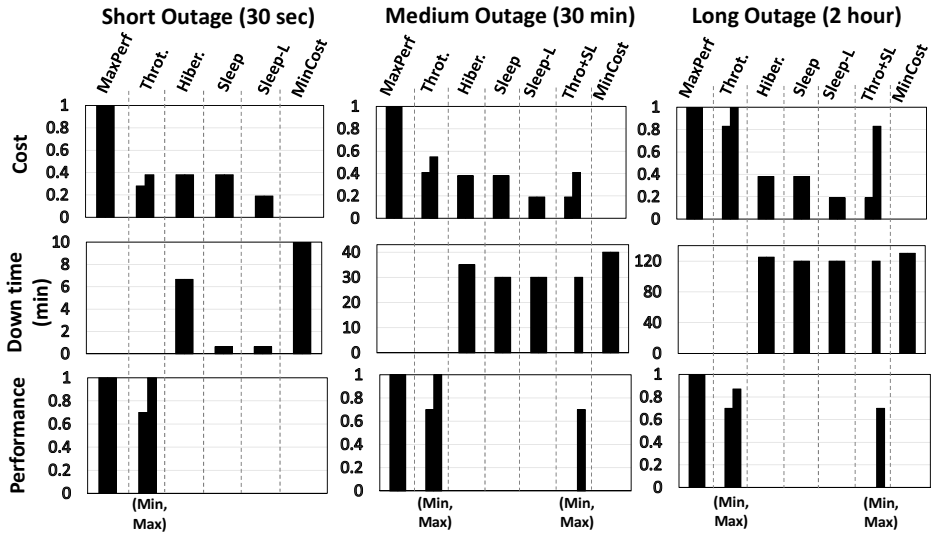


Figure 4.7: Tradeoffs for Web-search

fore 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 range for *MinCost* as shown in Figure 4.8. We find the tradeoffs between other techniques very similar to that of Specjbb application.

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

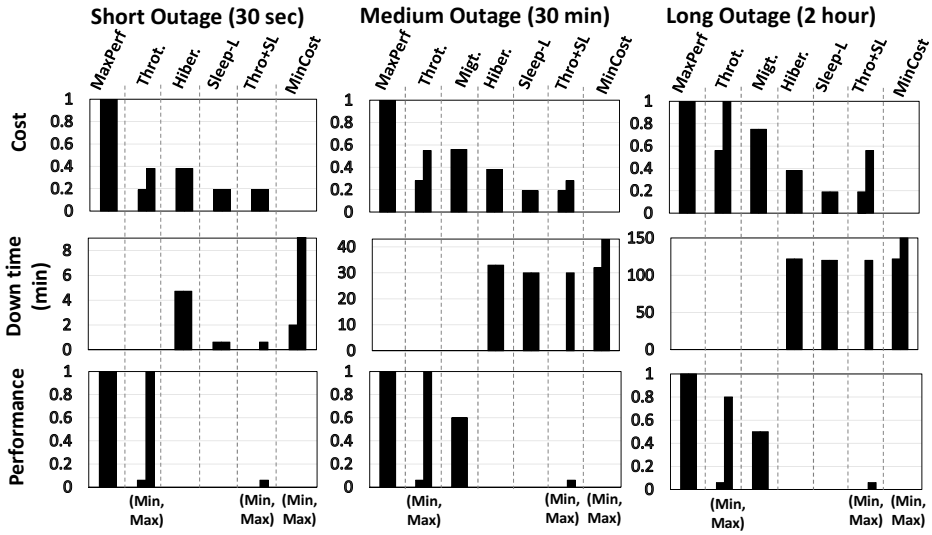


Figure 4.8: Tradeoffs for SpecCPU (mcf*8)

techniques such as *Hibernation* and *Migration*.

Chapter 5

Related Work

Power Capping/Demand Shaping: The impact of peak power on provisioning and heat load has drawn a lot of recent attention. This problem has been looked at from both provisioning for IT equipment - maximizing the utilization of the datacenter power distribution network [62, 36, 35, 45, 86] as well as the cooling cost viewpoint [91, 4]. There have been a number of control techniques to cap peak power consumption. Broadly, there are three primary categories of power capping knobs which have both cap-ex and op-ex benefits. Considering the power distribution network as a hierarchy flowing from incoming utility lines, to step-down transformers, UPS units, and Power Distributions Units, that subsequently feed to chassis and racks, and finally to individual outlets for each server, power capping knobs can be employed at one or more of these levels in the hierarchy.Temporal knobs include load scheduling or deferral, which temporally move portions of the load from peaks to valleys to shave the former. These may be implemented through dynamic voltage-frequency scaling (DVFS) techniques that slow down the execution, admission control techniques that drop load during peak demand, or by delaying execution to a low demand time (valleys) [66, 89, 114, 41, 115, 37, 73, 96, 13, 106, 14, 116]. Spatial knobs leverage heterogeneity of power demands at any time across the datacenter, and either statistically multiplexing their low probability simultaneous occurrence to co-locate them within a level [91, 45, 86, 40, 36, 62, 35, 42] or migrate workloads to regions in the hierarchy with headroom (valleys) from regions that are operating at their peak [45, 42, 61, 22, 14]. A recent set of knobs leverages energy storage devices (ESDs), such as batteries to provide just-in-time extra capacity for the power peaks by hoarding the required capacity (energy) in previous valleys

(when demand was lower) [46, 48, 59]. Depending on where they are placed in the power distribution hierarchy, ESDs can suppress peaks from propagating higher up in the hierarchy.

Power Virtualization: Unlike other resources, virtualization of power has not been studied extensively. Relevant work in this area is on accounting and control of power/energy usage of workloads in a co-hosted environment (e.g. [10, 100]) or within a multicore server [97], and coordinating (possibly conflicting) power management decisions (using power states, scheduling, etc.) of individual virtual machines co-hosted on a physical server [82, 100]. However, our work looks at (i) virtualizing entire power hierarchies and not just a single server, and (ii) looks to leverage multi-level energy storage (batteries) to achieve this goal (not just computing knobs). As we will see, sharing and isolation of battery usage across applications, especially in a hierarchical setting, introduces several new considerations in the power hierarchy management. Virtualizing and managing batteries across applications has been mainly studied (e.g. [19, 119]) in the context of embedded or mobile devices, where the primary goal is to prolong battery runtime, rather than power capping.

Backup Infrastructure Costs: Specific solutions in this area have mainly studied varying the redundancy and placement configurations [47, 69] of the backup equipment, to derive different availability-cost options, popularized by the famous *Tier* [102] 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, pro-active check-pointing and logging can be used (e.g. [26, 58, 16, 84]) to save or replicate application state and resume from the saved state. Intermittently available power has been explicitly considered in [98], 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 [81].

Chapter 6

Concluding Remarks and Future Work

6.1 Summary of Contributions

In this dissertation, we propose both theoretical framework, system implementation and evaluation in addressing issues on provisioning and harnessing ESDs for better demand response capabilities and cost-effective backup power infrastructure.

Energy Storage for Enhancing Demand Response (Chapter 3)

Provisioning and Control Methodology (Section 3.1): With a plethora of energy storage options, intricacies in their characteristics, pros and cons of placing them in different layers of the power hierarchy, and their suitability to diverse workload characteristics, there are numerous design choices when provisioning and dispatching power from these energy storage devices. We have presented three approaches on energy storage technologies/characteristics impacting their provision and operation in the datacenter: (i) a simple heuristic based on power trace characteristics for an quick and easy analysis on ESD provisioning [111, 110]; (ii) a simple analytical model to gauge the suitability of a given technology to different workload power profiles, and used this to identify the regions where each becomes cost-effective [113]; (iii) a systematic framework for capturing the different intricacies of ESD operations, and developed a generalized optimization platform for ESD placement and control in the datacenter. This platform can

be invaluable in datacenter design, capturing a whole spectrum of costs, constraints and workload demands [113].

System Software Support (Section 3.2): In addition to the theoretical framework, we have also investigated systems support for managing ESDs. As power capacities defined by cap-ex and/or op-ex constraints get more and more stringent, treating power as a first class resource in datacenter management becomes essential. Towards this, we have presented the design and implementation of vPower, virtualizing ESDs and all power equipments in datacenter hierarchy, to create, allocate and manage these devices for co-existing datacenter applications [112].

Energy Storage for Underprovisioning Backup Power Infrastructure (Chapter 4)

Design Space Exploration of Backup Infrastructure (Section 4.1): 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 [109].

Evaluation of Handling Power Outages with an Underprovisioned Backup Infrastructure (Section 4.2 and Section 4.3): 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 [109].

6.2 Future Work

Our work make notable steps towards provisioning and harnessing ESDs for improving demand response capabilities of datacenters. Nonetheless, there are several interesting directions to enhance and advance our current work.

Provisioning and Control Methodology Enhancements

Stochastic Formulations and Online Control: The provisioning and control offline optimization framework can be improved to deal with various forms of uncertainty bound to exist in the overall system - both in the workload, and the effect of control on performance and power. We can leverage existing work on (i) workload prediction, to derive statistical online models of power demands, and (ii) performance analysis, to derive performance models based on resource allocation and the employed computing control knobs. Stochastic formulations will then be used to re-cast our optimization framework. The system is still likely to deviate from such predicted stochastic behavior in the online setting. We will build upon techniques such as Markov Decision Processes (MDPs) to develop online control strategies. MDPs, however, suffer from the well-known curse of dimensionality, which limits their scalability. We will explore compaction techniques such as the Karhunen-Loéve expansion to perform a coordinate transformation to reduce the complexity of the underlying Markov chain. Concurrently, we will also explore the design and efficacy of highly scalable online control policies based on the theory of Lyapunov optimization.

Health-conscious ESD Dispatching Optimization: One of the biggest concern for using ESD for power capping/shaping is ESD health degradation due to extra charge/discharge cycles. Even though our provisioning and control framework captures battery state of health with a simple ESD health model, we will investigate more complex and hence more accurate ESD health degradation models and evaluate how state of heath affects ESD provisioning and control decisions. We will develop a multi-objective optimization theoretical framework which targets at minimize both cost and ESD health degradation. We will compute the Pareto optimal curve to study the trade-offs between different objectives. The impact of workload variations, ESD sizes and ESD control strategies on health degradation will be evaluated based on this framework.

System Software Enhancements

Application Utility Function Aware ESD Management: In addition to application specifying power demand hints, we will explore opportunities provided by application utility/quality functions. Charging and discharging ESDs based on these functions may help applications achieve better performance/quality/fairness and energy proportionality/efficiency. These utility functions also help system software to make better decisions in when and how much to tap into energy storage resource for real-time scheduling. We will leverage strategies in other domains such as grain, oil stockpiling and market based mechanisms and policies to incorporate utility function aware ESD management into our system software.

Boosting Demand Response Computing Knobs with ESDs: Our system software (vPower) integrates ESDs with IT power capping knobs based on power emergency duration and ESD SoC derived from application power demand hints. We will further investigate the design space of complementary interactions between energy storage and computing knobs in an online fashion (without future knowledge): (i) Exploit different ESD drain rates and the corresponding performance consequences when coupled with DVFS/throttling power state knobs. We will develop an optimization strategy to dynamically modulate DVFS while simultaneously deciding whether and how much to drain from ESDs. (ii) Study the placement/co-location of applications based on their power demand burstiness/correlations and ESD power/energy density across the power hierarchy. (iii) Explore migration related issues in a multi-layer ESD power hierarchy, i.e., what applications to migrate, when to migrate and where to migrate. We will build both optimization framework and online strategies encompassing DVFS/throttling, ESD drain rates, migration cost, applications performance and power demands, and ESD properties.

Energy Storage in Geo-distributed Datacenters and Smart Grids

By leveraging our methodologies and results for provisioning and harnessing ESDs in a single Datacenter, we can extend our work to multiple geo-distributed datacenters and even smart grids for better load redistribution, supply-demand matching, renewable energy penetration, etc. We will develop framework to address provisioning, dispatching, metering and communicating between these ESDs in distributed datacenters/grid based

on load demand, redistribution cost, renewable source variations as well as ESD states and characteristics at different locations. Further, our system software will allow ESDs at different locations be treated as “virtual energy storage”. Mechanisms and policies for managing these virtual energy storage in a much more distributed context will be investigated.

Demand Response vs. Power Backup

Until now, we have considered re-purposing existing UPS unit batteries for demand response. It is not clear if such dual usage - handling power outages and demand response - is the most effective option for the batteries in UPS units since the needs, costs, availability and health degradation considerations could be very different. We will carry out a detailed investigation of the design space of choices for provisioning and operating batteries for these dual purposes - separate batteries for each purpose, common pool of batteries for both purposes, and soft-reservations in this pool for the individual purposes with possible re-purposing dynamically based on demand. Such a detailed study would help identify the right way of provisioning (capacity) and purposing these batteries for the two intended purposes in the datacenter towards reducing cost, while meeting availability mandates.

Bibliography

[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] `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] F. Ahmad and T. N. Vijaykumar. Joint Optimization of Idle and Cooling Power in Data Centers While Maintaining Response Time. In *Proceedings of the Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2010.

[5] 2013. `http://www.apc.com/tools/isx/tco/index.cfm`.

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

[7] M. F. Arlitt and C. L. Williamson. Internet Web Servers: Workload Characterization and Performance Implications. *IEEE Trans. Netw.*, 5(5):631–645, 1997.

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

[9] 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.

[10] F. Bellosa, A. Weibel, M. Waitz, and S. Kellner. Event-Driven Energy Accounting for Dynamic Thermal Management. In *Workshop on Compilers and Operating Systems for Low Power (COLP)*, 2003.

[11] 2006. http://energy.gov/sites/prod/files/oeprod/DocumentsandMedia/MTS_Report_%to_Congress_FINAL_73106.pdf.

[12] D. Bhandarkar. Watt Matters in Energy Efficiency, Server Design Summit, 2010.

[13] A. A. Bhattacharya, D. Culler, A. Kansal, S. Govindan, and S. Sankar. The Need for Speed and Stability in Data Center Power Capping. In *Proceedings of the IEEE International Conference on Green Computing (IGCC)*, 2012.

[14] R. Bianchini and R. Rajamony. Power and Energy Management for Server Systems. *IEEE Computer*, 53(8), 2004.

[15] 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.

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

[17] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *Proceedings of ISCA*, 2000.

[18] California ISO Open Access Same-time Information System Hourly Average Energy Prices, Nov. 2011. http://oasishis.caiso.com/.

[19] Q. Cao, D. Kassa, N. Pham, Y. Sarwar, and T. Abdelzaher. Virtual Battery: An Energy Reservation Abstraction for Embedded Sensor Networks. In *Proceedings of IEEE Real-time Systems Symposium (RTSS)*, 2008.

[20] J. Chase, D. Anderson, P. Thakur, and A. Vahdat. Managing Energy and Server Resources in Hosting Centers. In *Proceedings of the Symposium on Operating Systems Principles (SOSP)*, 2001.

[21] J. Chase and R. Doyle. Balance of Power: Energy Management for Server Clusters. In *Proceedings of the Hot Topics in Operating Systems (HotOS)*, 2001.

[22] 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 USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2008.

[23] H. Chen, T. N. Cong, W. Yang, C. Tan, Y. Li, and Y. Ding. Progress in Electrical Energy Storage System: A Critical Review. *Progress in Natural Science*, 19(3), 2009.

[24] 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.

[25] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking Cloud Serving Systems with YCSB. In *Proceedings of the ACM Symposium on Cloud Computing (SoCC)*, 2010.

[26] 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.

[27] 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.

[28] G. Dhiman, G. Marchetti, and T. Rosing. vGreen: A System for Energy-Efficient Management of Virtual Machines. *ACM Transactions on Design Automation of Electronic Systems*, 16(1):6:1–6:27, 2010.

[29] K. C. Divya and J. Stergaard. Battery Energy Storage Technology for Power Systems - An Overview. *Electric Power Systems Research*, 79(4), 2009.

[30] Duke Utility Bill Tariff. `http://www.duke-energy.com/pdfs/scscheduleopt.pdf`.

[31] C. S. Ellis. The Case for Higher-Level Power Management. In *Proceedings of the Workshop on Hot Topics in Operating Systems (HotOS)*, 1999.

[32] EnerNOC Demand Response: Reducing Peak Load in Data Centers. `http://www.datacenterdynamics.com/ME2`.

[33] Facebook Open Compute Project, Nov. 2011. `opencompute.org`.

[34] Facebook Rack-level UPS for Improved Efficiency. `http://www.datacenterknowledge.com/archives/2011/04/07/facebook-unveils%-custom-servers-facility-design/`.

[35] X. Fan, W.-D. Weber, and L. A. Barroso. Power Provisioning for a Warehouse-Sized Computer. In *Proceedings of ISCA*, 2007.

[36] W. Felter, K. Rajamani, C. Rusu, and T. Keller. A Performance-Conserving Approach for Reducing Peak Power Consumption in Server Systems. In *Proceedings of the International Conference on Supercomputing (ICS)*, 2005.

[37] M. E. Femal and V. W. Freeh. Safe Overprovisioning: Using Power Limits to Increase Aggregate Throughput. In *Workshop on Power-Aware Computer Systems (PACS)*, 2004.

[38] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi. Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware. In *Proceedings of ASPLOS*, 2012.

[39] J. Flinn and M. Satyanarayanan. Managing Battery Lifetime with Energy-aware Adaptation. *Transaction on Computer Systems (TOCS)*, 2004.

[40] 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.

[41] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy. Optimal Power Allocation in Server Farms. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2009.

[42] L. Ganesh, J. Liu, S. Nath, G. Reeves, and F. Zhao. Unleash Stranded Power in Data Centers with RackPacker. In *Workshop on Energy-Efficient Design (WEED)*, 2009.

[43] A. Gavrilovska, K. Schwan, H. Amur, B. Krishnan, J. Vidyashankar, C. Wang, and M. Wolf. Understanding and Managing IT Power Consumption: A Measurement-Based Approach. In *Energy Efficient Thermal Management of Data Centers*. Springer, 2012.

[44] Google Server-level UPS for Improved Efficiency. `http://news.cnet.com/8301-1001_3-10209580-92.html`.

[45] 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.

[46] 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.

[47] 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 Power Aware Computing and Systems (HotPower)*, 2011.

[48] S. Govindan, D. Wang, A. Sivasubramaniam, and B. Urgaonkar. Leveraging Stored Energy for Handling Power Emergencies in Aggressively Provisioned Datacenters. In *Proceedings of ASPLOS*, 2012.

[49] S. Govindan, D. Wang, A. Sivasubramaniam, and B. Urgaonkar. Aggressive Datacenter Power Provisioning with Batteries. *To Appear in ACM Transactions on Computer Systems (TOCS)*, 2013.

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

[51] C. Isci, G. Contreras, and M. Martonosi. Live, runtime phase monitoring and prediction on real systems with application to dynamic power management. In *Proceedings of International Symposium on Microarchitecture*, pages 359–370, 2006.

[52] R. Jejurikar and R. Gupta. Energy Aware Task Scheduling with Task Synchronization for Embedded Real-Time Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6), 2006.

[53] 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.

[54] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. Bhattacharya. Virtual Machine Power Metering and Provisioning. In *ACM Symposium on Cloud Computing (SOCC)*, 2010.

[55] K. Kant. Data Center Evolution: A Tutorial on State of the Art, Issues, and Challenges. *Computer Networks*, 53(17), 2009.

[56] K. Kant. Supply and Demand Coordination in Energy Adaptive Computing. In *Proceedings of the International Conference on Computer Communications and Networks (ICCCN)*, 2010.

[57] A. Kejariwal, S. Gupta, A. Nicolau, N. Dutt, and R. Gupta. Energy Efficient Watermarking on Mobile Devices using Proxy-based Partitioning. *IEEE Transactions on Very Large Scale Integration Systems*, 14(6), 2006.

[58] 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.

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

[60] J. G. Koomey. Growth in Data Center Electricity Use 2005 to 2010, 2011. http://www.analyticspress.com/datacenters.html.

[61] 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.

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

[63] B. Li, G. Y. Keung, S. Xie, F. Liu, Y. Sun, and H. Yin. An empirical study of flash crowd dynamics in a p2p-based live video streaming system. In *Global Telecommunications Conference (GLOBECOM)*, 2008.

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

[65] X. Liu, P. Shenoy, and M. Corner. Chameleon: Application Level Power Management with Performance Isolation. In *Proceedings of the ACM International Conference on Multimedia*, 2005.

[66] K. Ma, X. Li, M. Chen, and X. Wang. Scalable Power Control for Many-Core Architectures Running Multi-threaded Applications. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2011.

[67] J. Mars, L. Tang, and R. Hundt. Heterogeneity in Homogeneous Warehouse-Scale Computers: A Performance Opportunity. *IEEE Computer Architecture Letters*, 10(2):29–32, 2011.

[68] M. R. Marty and M. D. Hill. Virtual Hierarchies to Support Server Consolidation. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2007.

[69] 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.

[70] Maxwell Ultracapacitor Uninterruptible Power Supply (UPS) Solutions. http://www.maxwell.com/products/ultracapacitors/industries/industry.asp%x?sid=UPS-SYSTEMS.

[71] S. McCluer and J. F. Christin. APC White Paper: Comparing Data Center Batteries, Flywheels, and Ultracapacitors. http://www.apcmedia.com/salestools/DBOY-77FNCT_R2_EN.pdf.

[72] D. Meisner, B. T. Gold, and T. F. Wenisch. PowerNap: Eliminating Server Idle Power. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2009.

[73] 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.

[74] 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.

[75] `http://www.memcached.org`.

[76] Memslap: Load Testing and Benchmarking a Server, 2012. `http://docs.libmemcached.org/memslap.html/`.

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

[78] Apr. 2011. `http://www.datacenterknowledge.com/archives/2011/04/25/microsoft-reveal%s-its-specialty-servers-racks/`.

[79] Microsoft Reveals its Specialty Servers, Racks, Apr. 2011. `http://www.datacenterknowledge.com/archives/`.

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

[81] 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.

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

[83] 2010. `http://www.inquirere.com/wp-content/uploads/2010/11/National-Survey-on-%Data-Center-Outages.pdf`.

[84] 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.

[85] M. K. Patterson, D. G. Costello, P. F. Grimm, and M. Loeffler. Data Center TCO; A Comparison of High-density and Low-density Spaces. In *Proceedings of the Conference of Thermal Challenges in Next Generation Electronic Systems (THERMES)*, 2007.

[86] S. Pelley, D. Meisner, P. Zandevakili, T. F. Wenisch, and J. Underwood. Power Routing: Dynamic Power Provisioning in the Data Center. In *Proceedings of ASPLOS*, 2010.

[87] E. Pinheiro, R. Bianchini, and C. Dubnicki. Exploiting Redundancy to Conserve Energy in Storage Systems. In *Proceedings of the Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2006.

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

[89] 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.

[90] D. V. Ragone. Review of Battery Systems for Electrically Powered Vehicles, 1968.

[91] 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.

[92] D. Rastler. Electricity Energy Storage Technology Options. Technical Report 1020676, Electric Power Research Institue, 2010.

[93] C. Ren, D. Wang, B. Urgaonkar, and A. Sivasubramaniam. Carbon-Aware Energy Capacity Planning for Datacenters. In *Proceedings of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2012.

[94] S. M. Schoenung and W. V. Hassenzahl. Long- vs. Short-Term Energy Storage Technologies Analysis. A Life-Cycle Cost Study. A Study for the DOE Energy Storage Systems Program. Technical Report SAND2003-2783, Sandia National Laboratories, 2003.

[95] 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.

[96] K. Shen, A. Shriraman, S. Dwarkadas, and X. Zhang. Power and energy containers for multicore servers. In *Proceedings of SIGMETRICS*, 2012.

[97] K. Shen, A. Shriraman, S. Dwarkadas, X. Zhang, and Z. Chen. Power Containers: An OS Facility for Fine-Grained Power and Energy Management on Multicore Servers. In *Proceedings of ASPLOS*, 2013.

[98] 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.

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

[100] J. Stoess, C. Lang, and F. Bellosa. Energy management for hypervisor-based virtual machines. In *Proceedings of the USENIX Technical Conference*, 2007.

[101] 2001. `http://www.onpower.com/pdf/EPRICostOfPowerProblems.pdf`.

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

[103] R. Urgaonkar, B. Urgaonkar, M. J. Neely, and A. Sivasubramaniam. Optimal Power Cost Management Using Stored Energy in Data Centers. In *Proceedings of the Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2011.

[104] A. Vasan, A. Sivasubramaniam, V. Shimpi, T. Sivabalan, and R. Subbiah. Worth Their Watts? - An Empirical Study of Datacenter Servers. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*, 2010.

[105] A. Verma, G. Dasgupta, T. Kumar, N. Pradipta, and R. Kothari. Server Workload Analysis for Power Minimization Using Consolidation. In *Proceedings of the Usenix Annual Technical Conference (USENIX)*, 2009.

[106] A. Verma, P. De, V. Mann, T. Nayak, A. Purohit, G. Dasgupta, and R. Kothari. Brownmap: Enforcing power budget in shared data centers. In *Proceedings of MIDDLEWARE*, 2010.

[107] VYCON: Flywheel Based UPS System in Data Centers. `http://www.vyconenergy.com/pq/ups.htm`.

[108] C. A. Waldspurger and W. E. Weihl. Lottery scheduling: Flexible Proportional-share Resource Management. In *Proceedings of USENIX conference on Operating Systems Design and Implementation (OSDI)*, 1994.

[109] D. Wang, S. Govindan, A. Sivasubramaniam, A. Kansal, J. Liu, and B. Khessib. Underprovisioning backup power infrastructure for datacenters. In *Proceedings of Internatonal Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2014.

[110] D. Wang, C. Ren, S. Govindan, A. Sivasubramaniam, B. Urgaonkar, A. Kansal, and K. Vaid. ACE: Abstracting, Characterizing and Exploiting Peaks and Valleys in Datacenter Power Consumption. In *Proceedings of the ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*.

[111] D. Wang, C. Ren, S. Govindan, A. Sivasubramaniam, B. Urgaonkar, A. Kansal, and K. Vaid. Ace: Abstracting, characterizing and exploiting datacenter power demands. In *Proceedings of IEEE International Symposium on Workload Characterization (IISWC'13)*, 2013.

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

[113] D. Wang, C. Ren, A. Sivasubramaniam, B. Urgaonkar, and H. Fathy. Energy Storage in Datacenters: What, Where, and How Much? In *Proceedings of SIGMETRICS*, 2012.

[114] X. Wang and M. Chen. Cluster-level Feedback Power Control for Performance Optimization. In *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA)*, 2008.

[115] 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.

[116] A. Wierman, L. L. H. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems: Optimality and robustness. *Perform. Eval.*, 69(12):601–622, 2012.

[117] 2009. `http://windows.microsoft.com/en-us/windows7/sleep-and-hibernation-frequ%ently-asked-questions`.

[118] 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.

[119] H. Zeng, X. Fan, C. Ellis, A. Lebeck, and A. Vahdat. ECOSystem: Managing Energy as a First Class Operating System Resource. In *Proceedings of the*

Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2002.

[120] F. Zhang, Z. Shi, and W. Wolf. A Dynamic Battery Model for Co-design in Cyber-Physical Systems. In *Proceedings of the International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 2009.

Vita

Di Wang

Di Wang was born and grew up in Jilin, China. He received B.E. in computer science and technology from Zhejiang University in 2005 and M.S. in computer systems engineering from the Technical University of Denmark in 2008. He joined the doctoral program in the Department of Computer Science and Engineering at Pennsylvania State University in January, 2010. During his doctoral program, he published in reputed conferences and journals in his research area that include ISCA, ASPLOS, SIGMETRICS, IISWC and TOCS. He received two Best Paper Awards and one Best Paper nomination. His work has been featured in CACM news and was chosen as IEEE sustainable computing register's pick of the month. He also won the Best Graduate Research Assistant award at Penn State. He worked for Teklatech (an EDA startup company in Copenhagen) as an R&D engineer in 2008, and interned at IBM Almaden research center in the summer of 2011 as well as Microsoft Research in the summers of 2012 and 2013.