DATA CENTER ENERGY MANAGEMENT

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Lakshmi Ganesh

January 2012

DATA CENTER ENERGY MANAGEMENT

Lakshmi Ganesh, Ph.D.

Cornell University 2012

Data centers form the underpinnings of the global technology revolution that is cloud computing. There is enormous pressure for data center growth and expansion, to meet the computational demands of an increasingly digital world. With energy costs overtaking server costs in data centers, energy is fast becoming a significant bottleneck to data center scale-out. Further, the global data center energy footprint is growing to be a significant burden on the world's energy resources. Yet energy is a signally ill-managed resource in most data centers; average data center energy efficiency is less than 50%. With increasing industry awareness of the magnitude and urgency of this problem, many solutions are cropping up to combat each of the several sources of data center energy inefficiency.

The objective of this dissertation is three-fold: First, we examine the causes of data center energy inefficiency from first principles, and identify the challenges involved in addressing them. We find two categories of energy inefficiency: *Idle resource energy consumption*, and *support infrastructure energy consumption*. Second, we present solutions to address each form of inefficiency. We describe two ways to combat idle resource energy consumption, and also present a systemic solution to tackle both forms of energy inefficiency. Finally, throughout this dissertation, we examine the related work and literature, and attempt to map them into the solution space to identify how the solutions relate with each other, and what gaps remain to be addressed.

The cloud has the potential to enable everything from ubiquitous computing and universal access to knowledge, to smart power grids, greater social connectivity, and

near-infinite extensibility of compute/storage power. The cloud turns computation into a utility, and by doing so, has the potential to make it accessible to a much larger part of the world. This dissertation explores ways to enable sustainable scaling of the data centers that power the cloud and enable this vision.

BIOGRAPHICAL SKETCH

Lakshmi was born in Trivandrum, in southern India, on the 27th of November, 1982, and grew up in nomadic fashion all over India. She has warm associations and close ties in most of India's metros and several little towns besides. Though she complained bitterly during the many moves of her childhood, in retrospect she wouldn't have it any other way.

Through the many variables in her life, Lakshmi has been fortunate to have the following constants: Her mother, Radha, who has managed the feat of being the best of parents as well as her best friend; her father, Ganesh, who has made his children the protagonists of all his stories; her brother, Anand, who has shown by demonstration the meaning of a good life; her sister, Meena, who has always had her back; and her extended family, which has unconditionally supported her. Lakshmi is named after her maternal grandmother, from whom she would like to learn the wonderful ability to evolve at every stage of life. Lakshmi has probably inherited her student gene from her mother, her writer gene from her father, and has learnt from her siblings about the pursuit of excellence. She is now learning from her husband and his family the secret of leading a life infused with joy.

On looking back, Lakshmi finds that most of her life's highpoints correspond to the times that she met and befriended some very special people. Her highschool years, spent in the company of friends who have grown to be foster-sisters, have the warm, golden glow of sepia photographs. Lakshmi's undergraduate years were truly formative: She found a home away from home among the many lovely people she met there; she discovered a love for theater; she learnt how to power through difficult times; she found her future husband, Madhav!

Lakshmi has now spent over seven years in the United States. It is a very broadening experience to find your place in a foreign country, and Lakshmi is in awe of what a

welcoming country this is. She spent two busy years in sunny Santa Barbara, getting her first taste of Systems research working with Ben Zhao. Ben gave her a gentle introduction to the process of crafting a research paper, and inspired her to pursue a doctorate degree. Lakshmi then found herself in gorgeous Ithaca, working with Ken Birman and Hakim Weatherspoon. Living in Ithaca is a little like doing a PhD—there are beautiful moments of discovery, mixed in with hard winters of deadline-chasing and doubt. Both are a trial by fire, and standing at the other end, Lakshmi feels the same mix of accomplishment and nostalgia for both.

Lakshmi now looks forward to post-doctoral work with Mike Dahlin and Lorenzo Alvisi at the University of Texas at Austin. Moreover, after six long years, Lakshmi is finally in the same city as Madhav, and she couldn't be more excited to begin the adventure of life together.

For my family.

ACKNOWLEDGEMENTS

People tend to picture PhD advisors as grim, bearded men with little interest in their students beyond their publishing ability. Picture instead an advisor who knows of your interest in theater and takes you to watch plays with him and his wife; an advisor who lets you pursue a Theater minor during your Computer Science PhD, absurd as the pairing seems; an advisor who shows you by example that it is possible to juggle a very successful career with a very successful family life. I have had not one but two advisors who have been friends as much as career mentors. I would like to thank Ken Birman and Hakim Weatherspoon for pushing me to work hard, but also urging balance; for pointing out my mistakes, but with humor; for showing me how to look for exciting questions to work on, but also be productively engaged while looking; and for much else besides. I have learnt all I know of the art of Systems research from them.

As I look back on my PhD, I see many points where thoughtful feedback and kind words have helped steer my course. I would like to thank Robbert van Renesse for being (possibly unbeknownst to him!) a stabilising influence in my career. I learnt some key lessons about precision in thought and speech, and time management from Fred Schneider (a pithy lesson in time management: work harder!). I admire Emin Gun Sirer for his remarkable work ethic, and his whole-hearted commitment to quality Systems work. I would like to emulate Bobby Kleinberg in his energy and enthusiasm as much as in the clear joy his work brings him. I admire Eva Tardos for being such a great teacher, and for being a role model for women in computer science. Finally, I shall remember David Feldshuh and Beth Milles fondly for welcoming me into the Theater world, and for showing me how many life skills may be learnt from the crafting of a play.

Most PhD careers, I think, have their ups and downs; mine was no exception. I had a great Ithacan community to celebrate my ups with me, and help me through the downs. My friends in the Distributed Systems group have inspired me with their exam-

ple, worked closely with me, given me key feedback for my work, and laughed with me over the many absurdities of PhD life. Thank you, Mahesh Balakrishnan, Tudor Marian, Yee Jiun Song, and Hussam Abu-Libdeh. My house and office mates were my Ithacan family. Ainur Yessenalina and Nikos Karampatziakis, I am going to miss our Sangam dinners very much. You have been very good friends to me. Sucheta Soundarajan, thank you for giving me a truly unique perspective on life, work, and animals! Saikat Guha, you have been a friend, a mentor, and an inspiration—thank you! As I look back I see so many lovely people who have enriched my life in Ithaca—Muthu Venkitasubramaniam, Rachel Lin, Edgar Velazquez-Armendariz, Bruno Abrahao, Nitin Gupta, Amit Sharma, and Parvati Iyer—thank you!

My non-Ithacan friends have supported me through my PhD career in innumerable ways—daily phone calls, visits, holidays together, and so much more. You are all so dear to me, and I miss you every day.

I come now to my best friend, Madhav. He also happens to be my husband. I want to thank him for making me laugh, for being dismissive of all my self-doubt, for seeing beyond my personal crises. His music, his worldview, his remarkable ability to see possibilities in everything enrich my life every day. Life is full of joy with such a traveling companion.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

From government bodies to private companies, banking institutions to hospitals, universities to individuals, data center use is ubiquitous. They are used to store and process private and corporate email and documents, financial and health records, media and games. They facilitate everything from data storage and computation, to communication, collaboration, and entertainment. Large (global-scale) organizations operate dozens of data centers spread across the world, each containing tens or even hundreds of thousands of servers. More data centers are springing up every day, and existing facilities are expanding continuously [74]. In both scale and reach, the global network of data centers is comparable to as basic an infrastructure as the electricity grid. Yet, in the matter of efficiency—of design and operation—the comparison fails.

The focus of this dissertation is data center energy management—a key aspect of data center operational efficiency. Energy costs can account for over 35% of the total cost of ownership (TCO) of data centers [78]. As servers grow ever more powerful, and data center server densities continue to increase, wattage per square foot has been growing as well. This compounds the amount of heat generated per square foot—in turn requiring the expenditure of more energy to remove. Energy costs now rival server costs [75], yet average industry energy efficiency is a mere 50% [61]. A telling measure of rising industry awareness of this problem is that global investment in greener data centers is projected to increase six-fold between 2011 and 2015, to $41B [8]. In this dissertation, we first establish the scope of the problem, and identify the fundamental research questions that underly it. We then describe our approach to addressing each of these questions.

1.1 Data Center Energy Management

Global-scale online services typically run on hundreds of thousands of servers spread across dozens of data centers worldwide [42]. Google is estimated to own over a million servers (as of 2009) [60], while Microsoft's Chicago data center alone is estimated to contain over 300,000 servers (as of 2011) [59]. These scales will only increase significantly as the cloud computing model [56] matures, and approaches what many perceive as a likely vision of the future—a handful of infrastructure providers hosting most of the world's data and computation [44]. As companies compete to take the lead in this space, the operational efficiency of their massive data centers assumes central importance; even small gains in efficiency translate into end-user perceivable cost reductions, providing key competitive advantage [48]. With energy costs comprising a significant portion of data center TCO, streamlining energy consumption has assumed central importance.

Data center energy consumption is also of concern due to its significant environmental impact. Studies show that the combined electricity use of the Internet and cloud (data center and telecommunications network) ranks fifth among the countries of the world, and is growing annually at a rate of about 12% [12]. Data centers worldwide are estimated to account for over 2% of global greenhouse gas emissions, and a significant fraction depend on coal for a majority of their power needs [12]. The cloud has the potential to significantly reduce global energy consumption, through enabling technologies like the smart grid, teleconferencing, and cloud storage, among others. Yet its growth, at current trends, will soon be bottle-necked by its own energy footprint. Committing to green data center design and operation is, therefore, essential to the realization of the cloud's rich potential.

Given the universal reach of data centers, and the truly global impact of their oper-

ational efficiency, it is perhaps surprising that average industry energy efficiency is less that 50% [61]. Before exploring the reasons underlying data center energy inefficiency, and identifying ways to mitigate it, it is worthwhile to take a closer look at how the data center industry got to this point. In what follows, we formally define the space, and present a brief history of the data center. We then describe the ideal data center energy profile, and proceed to identify a path towards achieving it.

1.1.1 Context and Background

A data center is a facility dedicated to housing a large group of networked servers and associated power distribution, networking, and cooling equipment, and used to host applications that store, manage, and process digital data. In essence, a data center is a warehouse-sized computer [35].

Physically, a data center may be a *brick-and-mortar* facility that typically takes years to build, and can house tens to hundreds of thousands of servers; Alternately, it may be a *containerized data center*, which is a shipping container pre-populated with a few thousand servers and associated infrastructure, and can be commissioned and deployed in a month or less [58].

Functionally, data centers may be *private*, where the entire facility is devoted to hosting applications belonging to the facility owner, or *shared*, where the facility owner leases out portions of the facility to different application providers [56]. A shared data center may be operated as one of three kinds of services, depending on the interface exposed to the lessees: *Infrastructure-as-a-Service (IaaS)* exposes the lowest-level interface, and is essentially the leasing out of physical servers; *Platform-as-a-Service (PaaS)* is one level higher, and leases out virtual CPUs and disks; and finally, *Software-as-a-*

Service (SaaS) leases out hosted software [56]. The scope of this dissertation includes brick-and-mortar, containerized, private and shared data centers.

The evolution of the data center can be traced back to the mainframes of the 1960's. These room-sized machines were very expensive, powerful, and highly customized for use in mission-critical data processing in large industries, military and space programs [46]. Commissioning one of these machines would often take years, and due to their highly complex and custom nature, developing applications for them was slow, error-prone, and lacked portability. The introduction of the personal computer (PC) in the 1970's suddenly made computers much more accessible, bringing them to small businesses and even homes [71]. Though not comparable to mainframes in terms of compute power, these machines were cheap, and much easier to develop software for with their standardized hardware and operating systems [46]. The next game-changer was the invention of Ethernet [57], and distributed computing. This led directly to the evolution of the modern data center, by enabling two parallel trends: First, the high-performance computing (HPC) community, which traditionally revolved around special computing hardware with unique processing capabilities (mainframes), found that the same massive computational and storage capacities could be achieved at much lower cost from large clusters of commodity machines [63]. Enterprises in all sectors, ranging from technology, to finance, military, and government, began to move away from the mainframe model, to the data center model. Second, personal computing increasingly shifted online, with data and computation both moving to remote third-party servers [81, 19]; the economies of scale argument saw online service providers concentrating their servers in large facilities, either private, or shared. Thus, in an interesting cyclical pattern, computing has evolved from the room-sized mainframe computers of the 1960's to the warehouse-sized server farms that comprise today's data centers.

The so-called *mega data center* is of relatively recent origin; data centers containing hundreds of thousands of servers did not exist even two years before the writing of this dissertation. This is a rapidly evolving space, with the pace of innovation not quite keeping up with the pace of growth/expansion. Data centers at large scale are, thus, often cobbled together from designs meant for a smaller, earlier generation operation [48]. The space is populated with point solutions and incremental improvements. What is needed is an examination of the fundamental sources of data center energy inefficiency. We start by identifying the energy profile of an ideally efficient data center; contrasting this with reality will then help uncover the research questions that need to be addressed in order to optimize data center energy consumption.

1.1.2 Energy Profile of an Ideal Data Center

Before delving into the sources of data center energy inefficiency, it is worthwhile to examine the energy profile of an ideal data center. This exercise establishes our goals in designing optimal energy management solutions.

Ideally, a data center should consume only as much energy as is needed to process incoming requests. Consider a request for a data item; given the application logic on the data center servers, this request translates to some number of CPU instructions, memory accesses, disk accesses, and network flows. These require some amount of energy to execute, which can be computed given the IT resource specifications of the data center. In reality, however, processing the request would additionally incur a number of energy overheads: energy used by idling resources, by air conditioners that cool the servers processing the request, and energy wasted in inefficient power delivery to the servers, among other overheads. The amount of these overheads depends on the energy

(in)efficiency of the data center. An ideal data center would minimize these overheads. We enumerate two target properties of a data center that capture this idea:

1. *Power-proportionality:* This property states that executing a given job consumes a minimal amount of compute energy, irrespective of how much time it takes—energy consumed by IT resources being proportional to work done [21]. This is possible only if base-line power consumption (i.e., power consumed when no job is being executed) is zero. In other words, idle resource power consumption must be zero.

2. *Power Utilization Efficiency (PUE) Close To 1:* This property states that the energy consumed during job execution is within a small margin of the amount of *useful* work done. In other words, the constant of proportionality relating energy consumed to useful work done should be close to 1. PUE is defined as the total energy consumed by the data center, divided by the total energy consumed by the servers in the data center. As energy consumed by a data center is divided between servers (which do directly *useful* work) and the power, cooling, and networking infrastructure that supports the correct functioning of servers (thus not directly useful), PUE can be much larger than 1. The industry average is 2 [78], which indicates considerable inefficiency in the operation of data center support infrastructure [77].

Together, these properties assert that the power consumed by a data center is a minimal function of its load. Data center power proportionality requires power consumption to track load, and eliminates overheads from idle resource power consumption. Low data center PUE ties power consumption closely to *useful* work done, and minimizes overheads from support equipment power consumption. Thus, a data center that is power

proportional and has a PUE of 1 would consume only as much energy as the application logic requires from the IT equipment. The next section examines the fundamental research challenges to achieving these target properties.

1.2 Research Questions

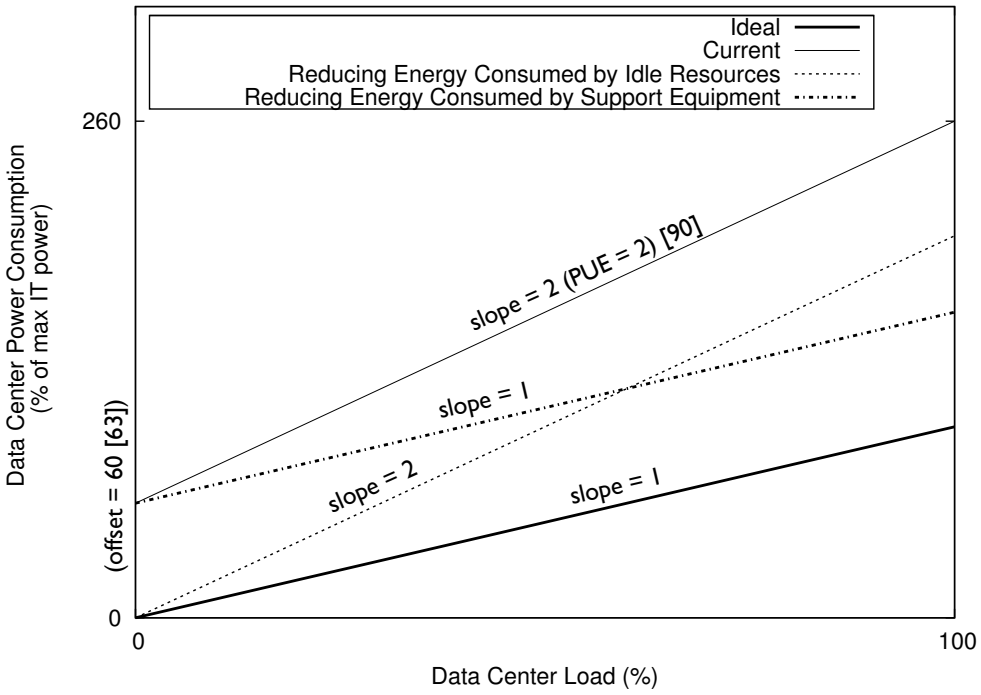


Figure 1.1: Schematic Diagram Of Data Center Power Consumption As A Function Of Load

Figure 1.1 compares an ideal power consumption curve, as described in the previous section, with the prevalent reality. The differences between these curves signal the presence of various inefficiencies in current data center design and operation. We identify two problem areas: (1) Energy consumption by *idle resources*, and (2) Energy consumption by *support equipment*. We discuss the magnitude of each problem area, and the challenges involved in addressing it.

1.2.1 Energy consumption by idle resources

The ideal data center consumes zero energy under zero load. The reality, however, is that in inadequately managed facilities, servers consume almost as much energy when idle or lightly loaded, as when heavily loaded [55]. This is the reason for the high offset in the power:load curve of the average data center (see figure 1.1). The problem is exacerbated by the fact that most data centers, being provisioned for peak rather than average load, are very lightly loaded on average—considerably less than 50% typically [48].

So why don't data center operators just turn off idle resources? Many server components also have the ability to operate in multiple power modes (corresponding to commensurate levels of performance), so that they can be manipulated to consume power proportional to their load, or desired level of performance. However, there are several challenges to this approach:

- *Performance Tradeoff:* Switching between power modes takes time, and can translate to degraded performance if load goes up unexpectedly. Most services can tolerate very little, if any, performance degradation.
- *Load Unpredictability:* The load on a given server can be impacted by a plethora of factors, including time of day, day of year, current world affairs, geography, and flash crowds, among others, making it very hard to predict accurately. Power managing servers without adequate fore-knowledge of their anticipated load can lead to significant performance degradation.
- *Short Idle Times:* Load spread—the set of servers that serve the load—can also vary continually (a result of load balancing, for example), leading to short idle times for most servers; this means that the time and energy cost of switching them to lower power modes is often not worth the potential energy saving from the

switch.

The research question here, then, is: *how can we minimize energy consumption by idle resources in a data center environment with high load variability, without unacceptable performance impact?*

There are two basic approaches to addressing this question. The first approach tries to reduce power consumption by idle resources by finding ways to enable switching them to lower power modes (or turning them off). This approach relies on designing mechanisms that improve the predictability and length of resource idle periods, to enable effective power-down. The second approach tries to eliminate (or reduce) the presence of idle resources at all, by provisioning fewer resources, and over-subscribing them. This approach is premised on the same principle as virtual memory—that all the contenders for the data center resources will not load-peak at the same time, thus allowing the pool of resources to be time-shared between them.

Examples of the first approach include disk power management solutions like Massive Array of Idle Disks (MAID) [31], Popular Data Concentration (PDC) [67], Hibernator [85], and Power-Aware Caching (PA-Cache) [84], among several others [40, 22, 41]. These attempt to reduce storage power consumption by powering down idle disks. Some of the shortcomings of these proposals are additional expense (MAID), inability to adapt to different workloads/applications (PDC), complexity (Hibernator), and insufficient returns (PA-Cache). Dynamic voltage and frequency scaling (DVFS) is a mechanism that allows CPU power to be manipulated to match its utilization. This is a useful tool, but needs an effective management framework that can maximize its benefit, by enabling sufficiently long idle CPU periods. We discuss how to design such a framework, in chapter 2.

Resource over-subscription solutions typically employ a power-tracking and capping approach [35, 51, 45, 25, 68, 79]. Power-tracking, as the name implies, is a mechanism to monitor power use, while power-capping prevents resources from exceeding a given (tunable) power cap. These are essentially safety mechanisms to enable resource over-subscription without the danger of overload and its repercussions. In chapter 3, we show how to go beyond power tracking and capping, to actively consolidate load to maximize resource utilization.

1.2.2 Energy consumption by support infrastructure

In addition to the servers and IT equipment that are doing directly useful work, data centers contain a considerable amount of support infrastructure like power distribution and cooling equipment, that enables the IT equipment to function correctly, but does not contribute directly to useful work done. In the ideal data center, the energy consumption of the support equipment should be a small fraction of the energy consumption of the IT equipment. In reality, however, support equipment consumes a comparable amount of energy to the IT equipment. This leads to the steep slope of the power:load curve of the average data center (see figure 1.1). Support infrastructure performs the following functions:

- *Cooling:* A prevalent rule-of-thumb in the industry suggests that for every Watt of energy being consumed by servers, about 0.5 W is needed to cool them [43]. Traditional data center cooling infrastructure consists of a chiller unit to chill the coolant used (water, or air), and fans to direct cool air towards the servers and hot air away from the servers. These are both intrinsically power-hungry processes. Further, the cooling infrastructure and the system it serves—the racks of variably

hot or cold servers—form a thermodynamically highly complex control system, which is hard to get exactly right [62]. The average data center, therefore, errs on the safe side and loads the cooling equipment more than required.

- *Power Delivery:* Power is typically delivered to a data center as high voltage AC power; this is stepped down to lower voltage AC power for distribution to racks for use by servers and other IT equipment. Inside this IT equipment, power supplies convert the AC power to the DC power needed for digital electronics. For every Watt of energy used to power servers, up to 0.9 W can be lost through this series of power conversions; further, more power is needed to cool the conversion equipment [76].
- *Power Backup:* In order to prevent outages, data centers use a backup power supply that can kick in temporarily if the primary supply fails. Traditionally, this backup takes the form of a central UPS (Uninterruptible Power Supply); power to the facility flows through the UPS, charging it, and is then routed to the racks. Significant power loss can result from this model, as the average UPS has an efficiency of only about 92% [36].

The research question here, then, is: *how can we minimize energy consumption by support infrastructure in a data center, without impacting correct server functioning or overall complexity?*

Solutions have been proposed, especially of late, to address each of the power overheads from support equipment. A highly effective solution to reduce cooling power overheads is *free cooling*, a system that uses ambient air for facility cooling, thus obviating the need for power-hungry chillers. It has been shown that free cooling can help bring data center PUE down to as low as 1.07 [11]. However, a severe limiting factor for this solution is the requirement that ambient temperatures be suitable for use in facil-

ity cooling—which does not hold for a majority of extant data centers. Power delivery efficiency has been shown to improve significantly by supplying the data center with DC power instead of AC power [76]. However, this shift also comes at a significant deployment cost. Finally, it has been demonstrated that moving from a central UPS power backup solution to a distributed model with each server backed up by its own battery can eliminate the power loss through UPS inefficiencies [36].

Another approach to reducing support infrastructure energy consumption is to power manage them in a similar manner to IT equipment—i.e., power them down when not needed. Along these lines, Thereska et al. [73] have shown how storage power consumption can be reduced by enabling the power down of entire servers, rather than just disks. However, to the best of our knowledge, explicitly power managing the power distribution, networking, and cooling infrastructure has not been tried. In chapter 4, we argue that this approach can be highly effective in reducing data center PUE, and show how to enable it without adding significantly to system complexity.

1.3 Contributions

In this dissertation, we present ways to address both of the research questions raised above. We describe two ways to reduce the energy wasted on idle resources, and achieve power proportionality. We also show how finer-grained control over the support infrastructure reduces its power burden, lowering data center PUE.

1.3.1 Energy consumption by idle resources

There are two ways to approach the problem of idle resource energy consumption. First, we attempt to match power consumption to load, by enabling power-down of idle resources. Second, we attempt to match load to provisioned power—we consolidate load so that resource utilization is maximized and idle resources minimized.

Matching Power to Load: In order to match data center power consumption to its load, we need to power down idle resources, so that baseline power consumption is minimized. The challenge here is in accurately predicting resource idle periods—if the idle period is not long enough, resource power down becomes counter productive. Current solutions have tried various means for predicting resource idle periods, with varying success. We suggest a new approach that circumvents the need for predicting resource idle periods, by manipulating load distribution intelligently.

We apply this approach in a low-power storage system design, called KyotoFS. KyotoFS is a distributed log-structured file system (LFS) [70] that leverages the read/write separation enabled by an append-only log model to significantly improve power-down opportunities among back-end disks. The key insight behind KyotoFS is that using a log makes all write accesses completely deterministic—they all go to the disks housing the log head. With write accesses constituting an increasing fraction of large-scale storage accesses [64], we find that a significant portion of disk accesses become completely predictable. Our evaluation suggests that KyotoFS can reduce storage energy consumption by up to 20%.

This approach finds a way to naturally distribute load in an energy-optimal manner, without the need for additional levels of indirection for load redistribution, or load analysis for predicting access distribution. At a high level, we can see this as a clean-slate

approach, rather than a patching approach. While both approaches have their place in engineering solutions, a clean-slate approach wins when its deployment cost is low, as it is here. As we discuss in chapter 2, the log model has regained prominence with the advent of flash storage; this allows us to combine the simplicity of a clean-slate design, with the low deployment overhead of an increasingly prevalent storage model (the log). As we will see, this combination is a recurrent theme in this dissertation.

Matching Load to Power: In order to match load to provisioned power, we need to consolidate load in such a manner that we maximize resource utilization. We propose a novel approach for consolidation: power-aware server placement in racks. The key insight behind this approach is the observation that there is considerable variation among the utilization patterns of servers in data centers; this suggests that cumulative load, per rack, could be smoothed by populating the rack with an intelligent choice of servers. A smooth load curve allows for maximal resource utilization.

RackPacker is an application of this approach. It is an algorithm for power-aware server placement on racks. It works by observing server utilization patterns over a period of time and determining optimal groupings of servers into racks such that average rack utilization approaches peak rack utilization, thus reducing resource stranding (idle resources). Our evaluation suggests that RackPacker can improve load consolidation to the extent that data center capacity could be increased by 18%. With more and more data center servers being virtualized, server placement decisions become a matter of virtual machine migration; thus, RackPacker can make frequent server placement decisions at low cost.

The RackPacker approach is applicable to more than just power consolidation. Data centers are increasingly being virtualized to improve server utilization; hosting multiple virtual machines on each physical server consolidates load, and reduces the physical

resources needed. Rackpacker is highly applicable in this context—it can be used to find near-optimal groupings of virtual machines to host on each physical server, to maximally consolidate load metrics such as CPU utilization, or memory usage. What is needed is an understanding of how this metric would aggregate over multiple virtual machines, which is non-trivial in some cases (memory, for instance) [47].

1.3.2 Energy consumption by support infrastructure

We show how to reduce support infrastructure energy consumption by moving to larger units of power management, such as racks, or even entire containerized data centers. The key insight behind this approach is that support infrastructure power consumption should be tied to the IT equipment it is meant to support; further, it is controlled in software, with the same algorithms controlling both the IT equipment, and its support infrastructure. The advantage of power management at rack granularity is that commodity racks are available that have their own power distribution, networking and cooling equipment; powering a rack down, therefore, also powers down its associated support infrastructure, thus improving PUE.

In our research on power-lean data centers, we show how to enable this approach, through power-aware data placement and load distribution. Further, we show that current data center design trends strongly support such an approach. Our evaluation results suggest that moving to rack-based power management can result in an eight-fold reduction in data center energy consumption, when compared with conventional power-management solutions.

The power-lean data center approach is another instance of combining a clean-slate approach with low deployment cost. The space of data center power management so-

lutions is fragmented—there are solutions for powering down idle IT equipment (disks, CPUs, servers), and solutions for reducing non-IT power overheads. Our solution unifies these two approaches; further, it does so at low cost by using prevalent data center practices such as data and compute cross-domain redundancy and rack-sized resource commissioning units (chapter 4).

1.4 Discussion: State of the Industry

Industry leaders like Google and Facebook have been making their data centers more energy efficient [83, 82, 65, 11]. In this context, it is reasonable to ask whether they have already solved the problems we describe. In a broader sense, we ask whether the industry is adequately addressing data center energy inefficiencies. We argue that while they are moving in the right direction, there remain important gaps to address.

In section 1.2, we sketched some of the solutions currently being proposed and deployed to streamline data center energy consumption; here we reiterate why they are not sufficient. Our argument is three-fold: first, the data center energy management space is fragmented—consisting of point solutions addressing individual sources of energy inefficiency, and lacking a systemic solution; second, most of the current solutions have high deployment overhead, and require significant data center design overhaul; and third, current solutions fail to address poor IT resource utilization in data centers, and the resulting power overheads from idle resources.

Google [82], Facebook [11], and others have reported facilities with PUE close to 1.00. They achieve this reduction of energy overheads through a number of point solutions. We have mentioned *free cooling*; this is arguably the single most effective means of reducing data center PUE. However, the required ambient conditions do not

pertain everywhere. With an increasing push for data center geo-diversification (originating from reasons of performance as well as failure-resilience) [39], it is likely that most future data centers also cannot assume suitable ambient conditions. Arguments for wave-powered data centers [34] face the same objections.

Other industry innovations for PUE reduction include per-server batteries [36], cold aisle containment [82], and a central CRAC controller [82]. Dell has designed servers capable of functioning in higher temperatures [10], thus reducing cooling needs. Future data center designs can and should make use of these solutions to improve their energy efficiency. However, existing facilities will need to undertake a significant design overhaul to adopt these solutions. In chapter 4 of this dissertation, we present a systemic data center energy management solution that has low deployment cost for both existing and new facilities.

Low data center server utilization is typically combated with virtualization. Hosting multiple virtual machines on each physical server can significantly increase average server utilization [9]. However, there are two weaknesses to this approach that we address in this dissertation. First, for optimal load consolidation, the right set of virtual machines need to be co-hosted in each server; co-hosting virtual machines with the same (or very similar) individual utilization curves will result in poorer load consolidation than if the co-hosted virtual machines have opposing utilization curves (i.e., when one curve peaks, the other troughs). Chapter 3 presents an algorithm to compute near-optimal sets of virtual machines for load consolidation. The second weakness to the virtualization approach is a more insidious one: most servers in a data center tend to have similar utilization curves, leading to poor overall load consolidation potential. No matter how many virtual machines you co-host in a physical server, if they all have load troughs at the same time, there will be idle server periods. Thus, virtualization and load

consolidation have to be complemented with idle resource management for true data center energy streamlining. Chapter 2 addresses the question of how to power down idle resources without impacting performance.

This dissertation argues that there is a need for a systemic approach to data center energy management, addressing both idle resource energy consumption and support infrastructure energy consumption. However, truly sustainable operations go beyond data center energy management to include investment in green energy research and development, usage of renewable energy, and active measures for protecting the environment; industry leaders are beginning to take heed [82].

1.5 Organization

We have discussed the importance of improving data center energy management, and identified two research questions that need to be addressed in order to do so. We have outlined our approach towards answering these questions, and contrasted them against related work.

The rest of this dissertation is organized as follows. Chapter 2 presents KyotoFS, a power-proportional storage system that matches power consumption to load by powering down idle disks. In chapter 3, we describe RackPacker, a server placement algorithm that consolidates data center load to match provisioned power, thus reducing resource stranding. Chapter 4 discusses how larger units of power management can drastically reduce support infrastructure energy consumption, resulting in improved data center PUE. Finally, chapter 5 presents some future research directions, and chapter 6 concludes.

CHAPTER 2

IDLE RESOURCES: MATCHING POWER TO LOAD

We have identified idle resource energy consumption to be one of the sources of data center energy inefficiency. Managing idle resources presents a tradeoff between performance and power; turning off idle resources saves power, but can result in a performance penalty or even service unavailability in case of accesses to the powered-off resources. This chapter shows how to walk the tightrope of reducing idle resource power consumption, while maintaining performance.

A significant fraction of the total cost of ownership (TCO) of data centers is the cost of keeping hundreds of thousands of disks spinning. We present a simple idea here that allows the storage system to turn off a large fraction of its disks, without incurring unacceptable performance penalties. Of particular appeal is the fact that our solution is not application-specific, and offers power-savings for a very generic data center model. In this chapter, we describe our solution, identify the parameters that determine its cost-benefit tradeoffs, and present a simulator that allows us to explore this parameter space. We also present some simulation results that add weight to our claim that our solution represents a new power-saving opportunity for large-scale storage systems. Finally, we demonstrate the practicality of our solution through a prototype implementation.

2.1 Context

The declining costs of commodity disk drives has made online data storage a way of life, so much so that companies like Google and Yahoo host hundreds of thousands of servers for storage [60]. However, a hundred thousand servers consume a lot of power! Not only does this translate to many millions of dollars spent annually on electricity bills,

the heat produced by so much computing power can be searing. Since disks account for a significant fraction of the energy consumed [85], several approaches for disk power management have been proposed and studied. We will examine some of these here. But first let us lay out some of the groundwork.

Any disk power management scheme essentially attempts to exploit one fact: disks can be run in high-power mode, or low-power mode, with a corresponding performance tradeoff. In the limit, a disk can be shut off so that it consumes no power. Given a large cluster of disks, only a fraction of them is accessed at any time, so that the rest could potentially be switched to a low-power mode. However, since mode transitions consume time and power, disk management schemes have to walk the tightrope of finding the right balance between power consumption and performance.

The solution space explored thus far in the literature can be divided as follows: (1) Hardware-based solutions, (2) Disk Management solutions, and (3) Caching solutions. Each of these solutions proposes a new system of some kind; *hardware-based solutions* propose novel storage hierarchy to strike the right balance between performance and power consumption; *disk management solutions* interject a new 'disk management layer' on top of the file system, which controls disk configuration and data layout to achieve power-optimal disk access patterns; *caching solutions* devise new power-aware caching algorithms that allow large fractions of the storage system to remain idle for longer periods of time, allowing them to be switched to lower power modes.

This chapter argues that there is a fourth niche as yet unexplored: (4) File System solutions. We do not present a new system; instead, we take an idea that has been around for well over a decade now—the Log-Structured File System (LFS) [70]—and argue that technological evolution has given it a new relevance today as a natural power-saving opportunity for large-scale storage systems. The key insight is that, where other

solutions attempt to predict disk access to determine which disks to power down, the LFS automatically provides a perfect prediction mechanism, simply by virtue of the fact that all write-accesses go to the log head. Section 3 explains and expands on this idea.

2.1.1 Idea Overview

To see why the LFS is a natural solution to the problem of disk power management, consider some of the challenges involved:

- Short Idle Periods: Typically, server systems are not idle long enough to make it worthwhile to incur the time and power expense of switching the disk to a low-power mode, and switching it back when it is accessed. This is a notable point of difference between server systems and typical mobile device scenarios (like laptops), which makes it hard to translate the solutions devised for mobile devices to server systems. The LFS localizes write-access to a small subset of disks; this feature, when combined with a cache that absorbs read-accesses, results in long disk idle periods.
- Low Predictability of Idle Periods: Previous studies [41] have shown that there exists low correlation between a given idle period's duration and the duration of previous idle periods. This variability makes it difficult to devise effective predictive mechanisms for disk idle times. The LFS neatly circumvents this problem by predetermining which disk is written to at all times.
- Performance Constraints: Server systems are often constrained by Service Level Agreements (SLAs) to guarantee a certain level of performance, so that finding a solution that provides acceptable performance to only a fraction of the incoming requests (albeit a large fraction) may often not be sufficient. The LFS provides an

application-independent solution that allows the system to perform consistently across a wide range of datasets.

- The law of large numbers: Large scale server systems process incredibly large request loads. Directing these to a small fraction of the total number of disks (the fraction that is in 'high-power mode') can significantly raise the probability of error and failure. The fact that the disks used in these contexts are typically low-end with relatively weak reliability guarantees exacerbates this problem. Our solution alleviates this problem by making sure that the live subset of disks is not constant.

The rest of this chapter is organized as follows: Section 2.2 describes some of the solutions explored in the first three quadrants mentioned above. Section 2.3 presents and analyzes our solution, while Section 2.4 discusses our evaluation methodology and results. We describe a prototype implementation in section 2.5, and conclude in Section 2.6.

2.2 Related Work

Hardware-based Solutions

The concept of a memory hierarchy arose as a result of the natural tradeoff between memory speed and memory cost. Carrera et al. point out in [22] that there exists a similar tradeoff between performance and power-consumption among high-performance disks and low-performance disks such as laptop disks. They explore the possibility of setting up a disk hierarchy by using high- and low-performance disks in conjunction with each other. In a related vein, Gurumurthi et al. [40] propose Dynamic Rotations Per Minute (DRPM) technology, whereby disks can be run at multiple speeds depending on

whether power or performance takes precedence. DRPM, however, poses a significant engineering challenge whose feasibility is far from obvious.

Another approach is proposed by Colarelli et al. in [31], using massive arrays of inexpensive disks (MAID). They propose the use of a small number of *cache* disks in addition to the MAID disks. The data in these cache disks is updated to reflect the workload that is currently being accessed. The MAID disks can then be powered down, and need only be spun up when a cache miss occurs, upon which their contents are copied onto the cache disks. This approach has several of the weaknesses that memory caches suffer, only on a larger scale. If the cache disks are insufficient to store the entire working set of the current workload, then 'thrashing' results, with considerable latency penalties. Further, the cache disks represent a significant added cost in themselves.

Disk Management Solutions

Pinheiro and Bianchini [67] suggest that if data is laid out on disks according to frequency of access, with the most popular files being located in one set of disks, and the least popular ones in another, then the latter set of disks could be powered down to conserve energy. Their scheme is called Popular Data Concentration (PDC) and they implement and evaluate a prototype file server called Nomad FS, which runs on top of the file system and monitors data layout on disks. Their findings are that if the low-access disks are powered down, this results in a considerable performance hit; they suggest instead that they be run at low speed. While their idea is sound, it is not clear whether this scheme would adapt to different workloads.

Son et al. propose another data layout management scheme to optimize disk access patterns [72]. Their approach uses finer-grained control over data layout on disk, tuning it on a per-application basis. Applications are instrumented and then profiled to obtain array access sequences, which their system then uses to determine optimal disk layouts

by computing optimal stripe factor, stripe size, start disk etc. Again, the wisdom of marrying the disk layout to the application seems questionable.

Hibernator, proposed by Zhu et al. [85], combines a number of ideas. It assumes multispeed disks, and computes online the optimal speed that each disk should run at. To minimize speed transition overheads, disks maintain their speeds for a fixed (long) period of time - they call this the coarse-grained approach. Hibernator includes a file server that sits on top of the file system and manipulates data layout to put the most-accessed data on the highest speed disks. The authors address the issue of performance guarantees by stipulating that if performance drops below some threshold, then all disks are spun up to their highest speed.

Caching Solutions

Zhu et al. [84] observe that the storage cache management policy is pivotal in determining the sequence of requests that access disks. Hence, cache management policies could be tailored to change the average idle time between disk requests, thus providing more opportunities for reducing disk energy consumption. Further, cache policies that are aware of the underlying disk management schemes (eg. which disks are running at which speeds, say) can make more intelligent replacement decisions. The authors present both offline and online power-aware cache replacement algorithms to optimize read accesses. They also show through experiments the somewhat obvious fact that for write accesses, write-back policies offer more opportunities to save power than write-through policies.

2.3 KyotoFS: A New Solution

We now argue that there remains an unexplored quadrant in this solution space. Caches are used to minimize accesses to disk. Good caching algorithms practically eliminate read accesses to disk. However, write accesses (whether synchronous or not) must still eventually access the disk. Thus, assuming perfect caching, disk access will be write-bound. Putting a disk management layer on top of the file-system to optimize data layout for writes is only halfway to the solution. To take this idea to its logical conclusion, it is necessary to rethink the file system itself. In the context of write-access optimization, a very natural candidate is the log-structured file system [70]. We now give a brief overview of the log-structured file system before describing the power-saving opportunity it represents.

2.3.1 Log-Structured File System

The Log-Structured File System (LFS) was motivated by a need to optimize the latency of write-accesses. Writing a block of data to a Seagate Barracuda disk costs about 11.5ms in seek time and 0.025ms/KB in transmission time. The key observation here is that seek time is a large and *constant* term in latency computation. To eliminate this term, LFS replaces write operations by append operations. Secondary storage is treated as a large append-only log and writes always go to the log head. Seek time is thus eliminated, and write latency becomes purely a function of the disk bandwidth.

How do reads work in the LFS? In the same way as in conventional file systems! Reads require random-access, and hence do not avoid seek-latency. However, the assumption is that with good caching mechanisms, reads will be a small fraction of disk

accesses.

As can be imagined, space reclamation is a tricky problem in log structured file systems. However, excellent solutions have been proposed to solve it, and one such is of interest to us: the disk is divided into large log segments. Once a log segment gets filled, a new log segment is allocated and the log head moves to the new segment. When some threshold of a segment gets invalidated, its valid data is moved to another segment (replacing that segment's invalid data), and it is then added to the pool of free log segments. Over time, this process results in a natural division of allocated segments into stable (ie.. consisting almost entirely of data that is rarely invalidated/modified), and volatile ones (which need to be constantly 'cleaned'). We will see how this feature can be used to save power.

2.3.2 LFS: A Power-Saving Opportunity

The disk-management policies described in the related works section essentially attack the problem by trying to predict in advance which disk any given access will go to. They optimize the data layout on disks to ensure that accesses are localized to some fraction of the disks, so that only these need be powered up. However, these are all probabilistic models - a new access has some probability of not fitting this model and needing to access a powered-down disk. Further, in such schemes, disk layout becomes tied to particular applications; two applications that have completely different access patterns might require different data layouts on disk leading to conflicts that reduce possible power-savings.

Since all writes in the LFS are to the log head, we know in advance which disk they will access. This gives us the perfect prediction mechanism, at least for write-accesses.

Besides being deterministic, this prediction mechanism is also application-independent. Thus, if most accesses to disks were writes, we could power down every disk but the one that the log head resides on. This, however, is an ideal case scenario. Our view is that, with a good caching algorithm (the power-aware caching algorithms described in section 2.2 are good candidates), reads to disk can be minimized, and only a small fraction of the disks need be powered on in order to serve all writes as well as reads.

However, what about the performance and power costs of log cleaning? Matthews et al present some optimizations in [54] to hide the performance penalty of log cleaning even when the workload allows little idle time. The power costs of log cleaning are a little more tricky to justify; however, this is where the natural division of segments into stable and volatile ones that the log cleaning process results in (as described above) can help. After a significant fraction of segments on a disk have been classified as stable, volatile, or free, we power the disk on and copy the stable segments to a 'stable' disk, volatile segments to a 'volatile' disk (disk is kept on), and the entire disk is freed for reuse. This is similar to the log cleaning scheme described in [66], which uses a 'hidden' structure embedded in the log to track segment utilization. Cleaning an entire disk amortizes the cost of powering it on.

The LFS has returned to prominence at the time of this writing as the file system of choice for use in flash storage devices [15]. Flash storage is increasingly being adopted in data centers as an intermediate level in the storage hierarchy between primary memory and disk. However, it has certain properties like block-sized erase and limited erase cycles, which make log-based storage models a good fit [15]. This means that an important piece of the puzzle for KyotoFS—log-based storage—is already seeing wide adoption in data centers today, facilitating industry adoption of KyotoFS.

Table 2.1: KyotoFS Evaluation: Sample trace characteristics

Number of accesses	476884
Number of files touched	23125
Number of bytes touched	4.22GB
Average number of bytes/access	8.8 KB

2.4 Evaluation

2.4.1 Methodology

We have proposed the use of the LFS in lieu of conventional file systems in data-center scenarios to achieve power conservation. For this idea to be accepted, two questions need to be answered in the affirmative: (1) *Does this new scheme result in significant power savings?*, and (2) *Does this new scheme provide comparable performance to existing schemes?* Further, the answers to these questions must be largely application-independent, and must apply to a generic data center model. To address these questions, we present a simulator - Logsim. Logsim consists of less than a thousand lines of Java code and is a single-threaded, discrete event-based simulator of a log-structured file system. Given a trace of read and write requests, Logsim returns the observed access latencies, disk utilization, cache-hit ratio, disk-mode transitions etc., for the chosen set of configuration parameters. We use real-world traces for our simulations from a web-server that serves images from a database [69]. Table 2.1 describes the characteristics of a sample trace.

The mechanism we simulate is as follows: All (non-empty) disks are assumed to begin in the 'on' state, and an access count (an exponentiated average) is maintained for each disk. The user specifies the maximum percentage (m) of disks that are kept powered on. Periodically (200ms, in our experiments), a 'disk check' process scans the access

count for each disk and powers down all but the most-accessed top $m\%$ of the disks, as well as any disk which does not have access count at least t. t is 0 in our experiments. If a cache-miss results in an access to a powered-down disk, then this disk is spun up (to remain powered on until the next 'disk check'), and there is a corresponding latency penalty. Judicious choice of m and t minimizes the probability of this occurrence.

2.4.2 Results

To save power, we must turn off some percentage of disks in the storage system. However, there are two opposing forces at play here. A large number of powered-on disks results in good performance (low latency), but also low power savings. On the other hand, decreasing the number of powered-on disks incurs two possible penalties: increased latencies, and increased mode-transitions. Mode-transitions consume power and thus counter the potential savings achieved by powered-down disks. To find the optimal percentage of disks to be powered down, we ran a set of simulations on Logsim and varied the number of disks that we kept powered up from none (except the log-head disk), to all, in steps of 20%. Thus, out of a total of 66 disks, 1, 13, 26, 39, 52, and 66 disks were kept powered up, respectively. For each run, we examine both its performance (in terms of observed access latencies), as well as its power-consumption. Figure 2.1, 2.2 and 2.3 show the results of these simulations.

The performance of our system depends heavily on its cache configuration. Since cache optimization is an orthogonal issue that comprises an entire field of research in itself, it is important to isolate its effect on performance. To achieve this, we implemented an 'ideal cache' algorithm, which we term the *oracle*. Experiments using the oracle approximate the best performance we could achieve since an oracle has future

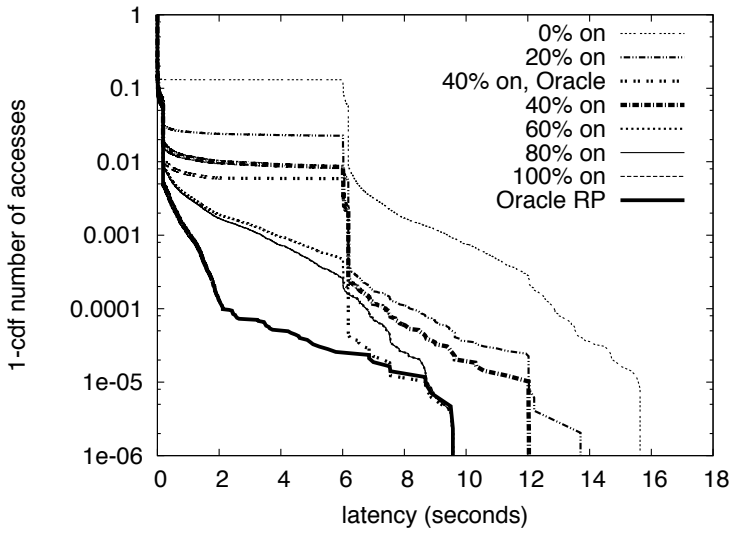


Figure 2.1: KyotoFS: Effect of varying percentage of powered-up disks on performance (complementary CDF).

knowledge and is able to replace items accessed furthest in the future [33]. In figure 2.1, 2.2, 2.3, the data points that use this algorithm are annotated with the word 'Oracle'.

Finally, we also wish to compare our system against conventional (not log-structured) file systems. As an approximation of such a system, we implemented a 'random placement' (RP) algorithm, which maps each block to a random disk. All disks are kept powered up, and ideal caching (oracle) is assumed. This data point is labeled 'Oracle RP' in our graphs.

Having set the context, let us examine our results. Figure 2.1 shows the relation between performance (per-access latency) and the number of disks that are powered on. If we imagine a line at y=.001 (ie.. 99.9% of the accesses live above this line) 60% disks ON is the third best configuration, next only to the Oracle RP and 100% disks ON configurations. Further, the performance degradation in going from 100% disks ON to 60% disks ON is negligibly small. The principal take-away is that, for the system under test, the optimal configuration is to have 60% of the disks powered on. In other words,

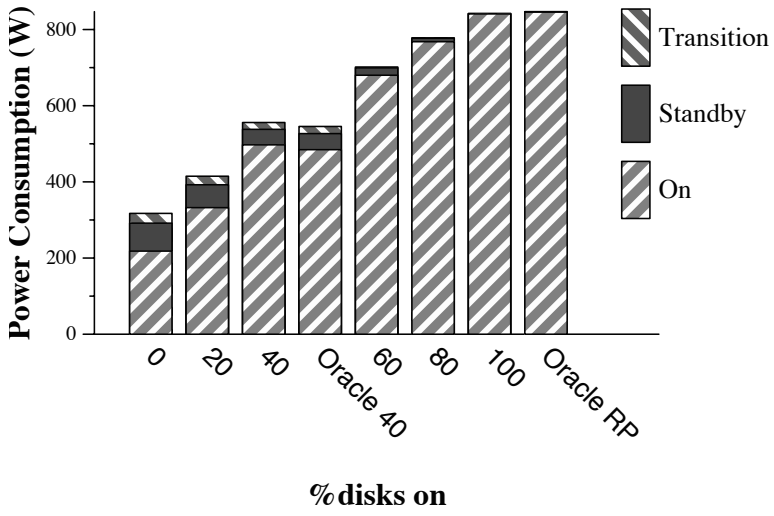
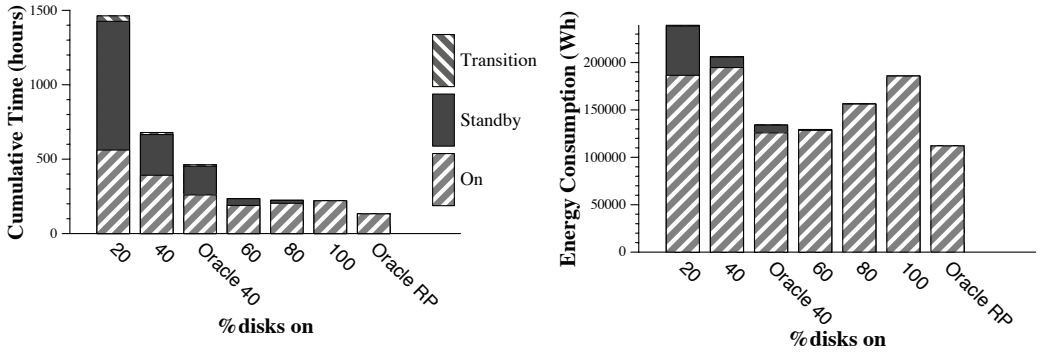


Figure 2.2: KyotoFS: Effect of varying percentage of powered-up disks on power consumption

40% of the disks can be spun down while still maintaining performance comparable to that of a conventional file system.

Figure 2.2 shows an estimate of the actual power savings achieved by our solution. The height of each bar is the average power consumed while processing the trace. Further, each bar shows the break-up of power consumed by powered-up disks (On), powered-down disks (Standby), and mode-transitions (Transition). We assume the following disk specifications: Average operating power = 12.8 W, standby power = 1.5 W, mode transition power = 13.2 W, and time for transition = 6s. We see that turning off 60% of the disks results in about 18% power savings while maintaining acceptable performance.

Figure 2.3(a) shows how much time the disks spend in on/standby/transition states. The height of each bar is the cumulative time spent by each disk in each of these three states. When 0% disks are on, the run takes 7253 cumulative hours; we omit this bar from our graph for clearer presentation. We see that both the total duration of the experiment as well as the number of mode-transitions increase as the percentage of disks that



(a) Trace running time (b) Energy

Figure 2.3: KyotoFS: Effect of varying percentage of powered-up disks on trace running time and energy

is powered on is decreased. Figure 2.3(b) shows the effect of this increased running time on energy consumption (we omit the bar for 0% disks on, as it corresponds to about 800 kWh and is off the scale of the graph). However, as in figure 2.1, we see that keeping 60% disks on strikes an acceptable balance. While there is an inherent tradeoff between power and performance as is illustrated in figure 2.3, we show in the next section that a less read-dominant workload can actually result in KyotoFS outperforming conventional file systems, while also consuming less power.

2.5 Prototype

In this section, we describe a proof-of-concept implementation of KyotoFS, and show that it is a practical storage solution which can match raw disk performance and even exceed it in certain cases.

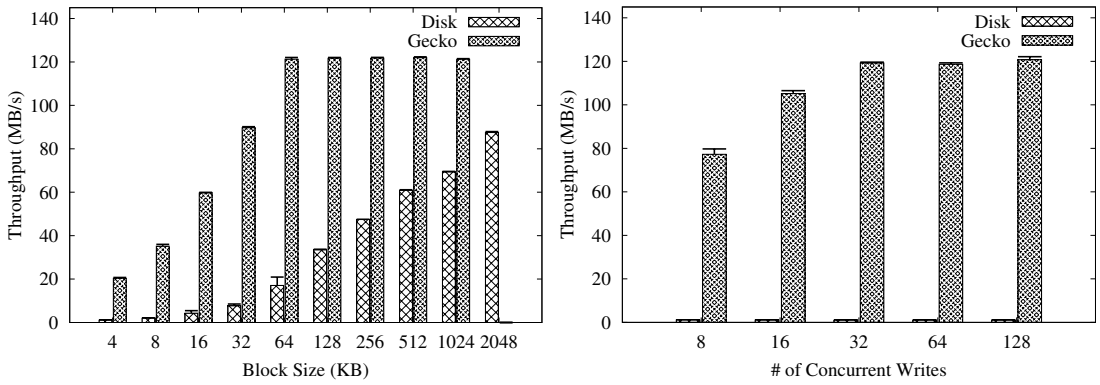
We implemented a prototype of KyotoFS as a block device driver called Gecko. Gecko is a non-distributed version of KyotoFS, running on a single node, and occupying the same position in the storage stack as conventional RAID solutions. It implements

a logical address space over a physical one that is formed by concatenating or chaining all the drives in the node's disk array into a single address space. In its simplest form, Gecko uses the physical address space strictly as a circular log. It maintains tail and head pointers that inform it of the current location of the log in this address space. It also maintains a blockmap from logical blocks to physical positions on the log. When updates are initiated by the OS on the logical address space, Gecko logs them to the tail and then updates its blockmap. When reads occur on the logical address space, Gecko checks this blockmap for the current position of the requested block in the log, and then fetches it from the disk array.

Each link in a Gecko chain can be a mirrored pair of drives for availability and read throughput properties similar to RAID-1. The advantage of this model is that one of each mirror pair on the log tail can be powered down without impacting data availability. This would lower read throughput, but would not compromise availability or fault tolernce. In this model, Gecko can also decouple log cleaning across mirrors in the log tail. Consider an example where the log tail consists of mirrored drives D0 and D0', and the log head consists of drives D1 and D1'. Gecko cleans D0 first, moving valid data to the head of the log at D1 and D1'.

The key power-saving opportunity in KyotoFS is from the separation of reads and writes to different disk drives, so that there is power-down opportunity for disks that are read-only (the disks in the log tail). Gecko ensures this separation by maintaining the log head and log tail on different drives. We are also exploring ways to send cleaning and first-class writes to different drives on Gecko, so that the write throughput of the system is unaffected by cleaning.

We evaluated Gecko performance on a node with six disk drives. Figure 2.4 shows that Gecko exceeds raw disk performance for writes; since Gecko sequentializes writes



(a) Varying Block Size (b) Varying Number of Concurrent Writers

Figure 2.4: KyotoFS Implementation: Random write throughput without and with Gecko

by design, this throughput gain is achieved at no cost to latency. Combined with the power savings demonstrated in the previous section, the improved write throughput illustrates the advantages of a log to power and performance.

2.6 Conclusion

In this chapter, we point out a new opportunity for saving power in large-scale storage systems. The idea is elegant in its simplicity: Log structured file systems write only to the log head; as a result, if read accesses are served by the cache, then write accesses touch only the log head disk, potentially allowing us to power down all the other disks. Existing solutions like disk management solutions and caching solutions are typically application-specific; our solution, on the other hand, is applicable to any cacheable dataset. Since existing solutions are typically layered on top of the file-system, they could be used in conjunction with our solution to take advantage of application-specific optimizations.

We provide simulation results to support our claim that power-savings are possible

using a log-structured file system. We also demonstrate the practicality of our solution through a prototype implementation. Our principal contribution here is in having shown a new fit for an old idea; we believe that the log-structured file system shows promise as a power-saving opportunity for large-scale storage systems.

CHAPTER 3

IDLE RESOURCES: MATCHING LOAD TO POWER

The previous chapter looked at ways to reduce idle resource energy consumption by enabling the power-down of idle resources. In this chapter, we adopt a complementary approach: we minimize resource idling through *load consolidation*. Load consolidation reduces resource idling, and allows maximal utilization of the data center resources.

The capacity of a data center is defined in many dimensions: power, cooling, space, water, network bandwidth, etc. Running out of resources in any of these dimensions means that the service provider needs to build or rent another data center to facilitate business growth. Among these resources, power is usually the first to be exhausted because of the load limitation on the power grid and the increasing power density of computing[1]. However, recent studies [52, 35] have found that the average data center's power resources are highly underutilized.

In this chapter, we look at ways to optimize the power utilization in data centers by addressing the following question: *How many servers can a facility with a given power capacity host?* In common practice, this number is arrived at by dividing the provisioned power capacity by the power rating of each server. This rating might either be the nameplate rating of the server (which is usually a substantial over-estimate), or—which is slightly better—the server's experimentally measured peak power consumption. However, both these schemes suffer from the weakness of all static provisioning solutions—they do not account for the variability of load on the servers and the resulting dynamics of their power consumption.

[1] defined as the amount of power consumed by a rack of servers occupying a unit space (e.g. square foot)

We propose an algorithm that studies the power consumption behavior of the servers over time, and suggests optimal ways to combine them in racks to maximize power utilization. At the heart of such a dynamic provisioning scheme is the following intuition: the actual power consumption of each server is not always (and often very rarely) equal to its peak; hence, by intelligent *over-subscription* of the provisioned power, we can unleash the stranded power to host more servers. In other words, if we employed this scheme to populate our facility, we would exceed its power capacity if all of the servers were running at peak load; however, since the probability of such an event is vanishingly small, we are (with very high probability) fine.

Our solution takes advantage of two technology trends in data center computing: 1) *virtualization*: the use of virtual machines (VM) to consolidate services and ease software migration; and 2) *power capping*: the ability to adjust the power state of a server to prevent it from exceeding a given power cap. With VMs, it is easy to move services among physical servers, so that "matching" servers can be placed together to reduce the probability of exceeding a power budget. With power capping, the rare events of exceeding power limits can be mitigated by reduction in performance. Although we still aim at minimizing the power capping probability, reaching or temporarily exceeding power capacity will not cause catastrophic failures.

With these assumptions, our algorithm—RackPacker—solves what we term the *server placement* problem: *Given actual power consumption profiles over a period of time for a set of servers, what is the least number of racks that they can be packed into without exceeding any rack's power cap?* A brute force optimization formulation can reduce this problem to *vector bin packing* [26], where d time instances of interest are d dimensions of an object and the bin size in each of the d dimensions is the rack power cap. However, in this formulation, d could be a few thousands if the provision-

ing cycle is a day and power samples are collected every 30 seconds. Since this vector bin packing formulation leads to an NP-hard problem with prohibitively many dimensions, we use a number of domain-specific optimizations to arrive at a near-optimal solution efficiently. One of the central insights we use is that some, but not all, servers' power consumption can be strongly correlated due to their application dependencies or load balancing designs. Hence, it is desirable to find servers that show anti-correlated, or strongly time-shifted behavior and pack them together to minimize power capping probability. Our experiments with RackPacker show from 15-30% improved efficiency in packing servers in racks. Note, however, that RackPacker provides a probabilistic solution—should server power consumption diverge significantly from the norm, rack capacity can be exceeded.

In Section 3.1, we describe the background and common practice on rack power provisioning and show the opportunity for unleashing stranded power. We then describe our algorithm—RackPacker—in Section 3.2. We discuss the evaluation of RackPacker in Section 3.3 and present related work in Section 3.4. Finally, Section 3.5 summarizes our findings and suggests interesting avenues for future work.

3.1 Stranded Power

To understand the rack packing challenges and opportunities, we first describe the power distribution and provisioning architecture in a typical data center. Power consumed by a data center is usually divided into *critical power*, which is UPS backed up and used by IT equipment, and *non-critical power*, which is used by cooling and other parts of the facility that do not require UPS backup. We only consider critical power utilization here.

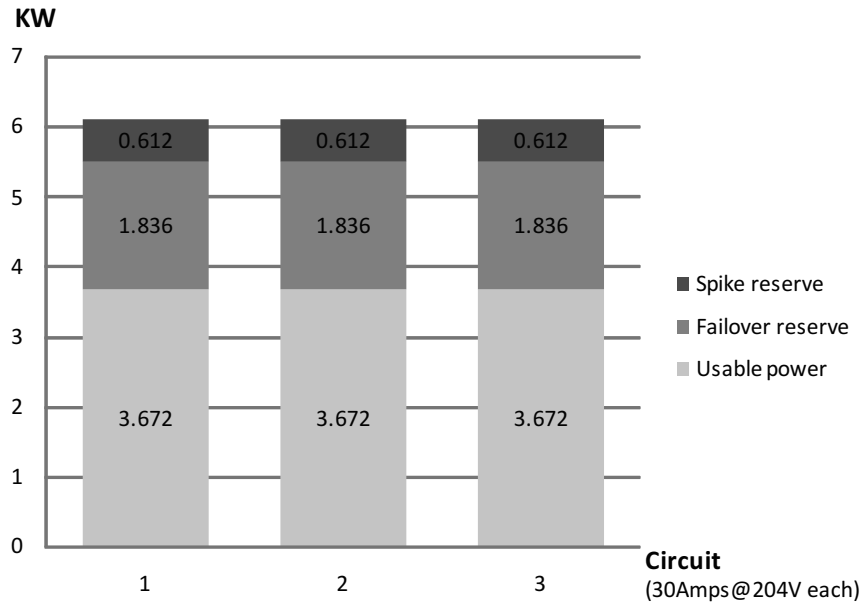


Figure 3.1: An illustration of power provisioning at the rack level. About 40% of available power is reserved for handling spikes and failover.

Critical power in a data center is delivered to remote power panels (RPP) in each server room (usually called server co-locations or colos), split into many circuits there, and then distributed to server racks in that colo. Every circuit has a defined capacity, and is regulated by a circuit breaker, which is the physical defense for catastrophic power failures. For redundancy purposes, a rack usually has multiple circuits, each in the form of a power strip. Servers, typically dual corded, spread their power load across the power strips they plug into. Figure 3.1 shows the power provisioning chart for a rack with 3 circuits, with power load evenly distributed over the circuits, (i.e. each server is plugged into two of the three power strips). There are two overheads that limit the amount of power usable by the servers: *spike protection* and *failover protection*. If each circuit is rated at single phased 30Amps and 208V, then the total available power at each circuit is 6.24KW[2]. However, 10% to 20% of the total power is reserved to handle spikes in

[2]Technically, it is 6.34KVA. For ease of discussion, we ignore the power factor and treat W and VA interchangeably.

the power grid or load (10% is shown in this plot). Furthermore, in order to support failover—in the sense that when one of the three power strips fails, all servers can safely use the remaining two power strips—another 30% of the total power has to be set aside. Thus, the *usable power* to the servers is only up to 60% of the total power—3.74 KW per circuit, or 11.2KW for the entire rack. In fact, this rather conservative power provisioning baseline encourages probabilistic over-subscription, since temporarily exceeding the power cap is likely to be safe.

The common practice of power provisioning, however, does not even fully utilize the 60% usable power. Server vendors usually give an estimated *nameplate* power consumption indicating the maximum possible power consumption of the server. For example, the power calculator tool from HP [3] rates 395W for a ProLiant DL360 G5 server, with two Xeon E5410 2.33GHz quad-core CPUs, four 2GB DIMM, and two 146.8GB SAS HDD. In other words, a 11.2KW rack can host at most 28 such servers, even though each server only occupies one unit in a typical 44 unit rack.

In reality, the nameplate power allocated to a server is never fully used. Using server profiling, one can arrive at a discounted power rating, which is lower than the nameplate power rating. For example, if the DL 360 server has never consumed more than 300W, using the discounted power rating, a rack can host 37 such servers.

Static power provisioning, even with discounted power rating, can still leave a large amount of power stranded. Figure 3.2 shows a power consumption trace over a day of a production server accessed by millions of users. We have two observations. First, server power consumptions change due to the load fluctuation. Slow and quasi-periodic load fluctuation has been observed in a lot of web traffic, including web sites [25] and instant messaging [27]. This fluctuation can become even more significant as idle power consumption is decreasing for newer servers. Secondly, in addition to the slow fluctua-

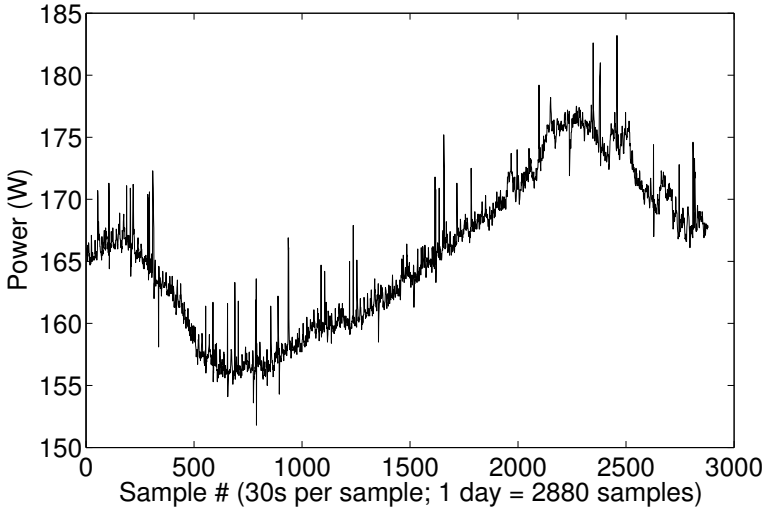


Figure 3.2: A real power consumption trace of a production server.

tion, there are spikes, caused by short term load variation such as scheduled processing intensive tasks or flash crowd visitors. The discounted power rating—being a worst case estimate—must include both the peak of the fluctuation and the highest spikes; thus it can be overly conservative.

Power over-subscription can take advantage of two dynamic properties of actual server power traces:

- Not all servers fluctuate to the peak at the same time. The usage patterns of on-line services can be diverse. For example, websites for financial news and services may reach their peak around late morning when both east and west coast customers are on-line and the stock market is open. However, home entertainment sites may reach their peak in the evening. If we can bundle services that are maximally out of phase, then the peak of the sum is less than the sum of the peaks.
- Servers that are managed by the same load balancer or have close dependencies can have strong correlations among their spikes. Statistically, placing services that

are *anti-correlated* will lead to smaller probability of their seeing simultaneous spikes.

These observations motivate us to design RackPacker, which statistically guarantees that over-subscribed sets of servers do not exceed rack level power caps.

3.2 The RackPacker Approach

3.2.1 A Running Example

Throughout the rest of this chapter, we use 831 servers from a popular on-line service as a running example for our discussion. Functionality-wise, these servers largely belong to three categories, which we call Types 1, 2, and 3. They are divided into several clusters, where each cluster is managed by a load balancer. Server workloads show strong correlations, because of both functionality dependencies and load balancing effects. For example, when there is a flash crowd, servers behind the same load balancer experience a power spike at the same time, while servers across load balancers are less correlated. Due to the nature of the application, we also observe that about 2 hours after servers of type 1 reach their peak workload, servers of type 3 reach their peak. The tight coupling among server tiers and the relatively high CPU utilization, reaching over 75% at peak load, make this a challenging set of servers for rack packing.

These servers have a nameplate power rating of 350W; based on this number, a 11.2KW rack can host 32 servers. In other words, we need 26 racks to host these servers in the most conservative situation.

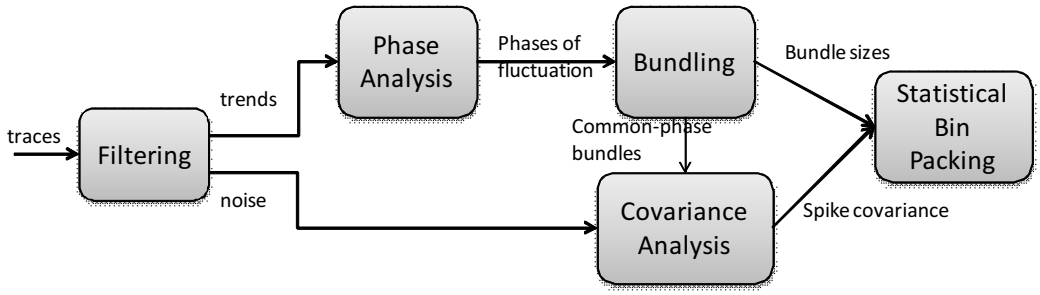


Figure 3.3: The flow of the RackPacker algorithm.

3.2.2 Rackpacker Overview

RackPacker takes a data-driven approach that uses collected power consumption traces to support server placement decisions. We assume that services are hosted by virtual machines, even though there may be only one VM per physical server. VMs enable fast re-positioning of services without moving the servers physically. This allows the server placement decisions to be made frequently—at a weekly or even daily basis—and aggressively. The RackPacker algorithm, thus, only needs to predict short term traffic growth. In the rest of this chapter, we use the terms server and service interchangeably. That is, a server of type 1 refers to a VM hosting service type 1 running on a physical server. We only consider homogeneous server hardware.

Figure 3.3 shows the key components in the RackPacker algorithm.

By profiling or monitoring a server operation, we model the server power consumption with a time series (rather than a single number). The time series is first filtered to obtain the low frequency power consumption baseline, and the high-frequency noise that captures spikes. The noise signal has zero mean. Its variance represents how "spiky" the transient power consumption can be. The goal of obtaining the low-frequency components is to identify the baseline fluctuations reflecting workload trends, specifically their phase. Using this phase information, we can sift through the servers and bundle

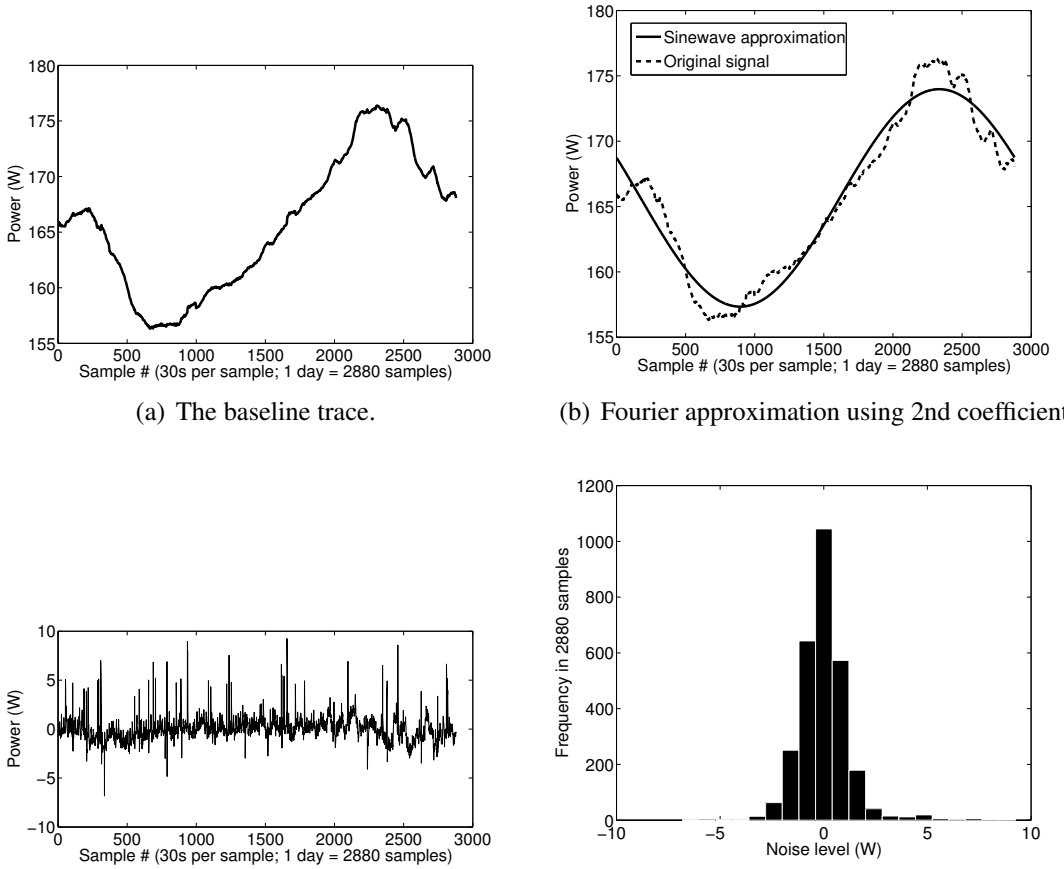
those that are most out of phase. The bundles are then treated as the unit for rack packing. The high-frequency noise goes through a covariance analysis that measures the likelihood that two bundles may have spikes at the same time. This statistical measure, together with the baseline of the bundles, is used in a statistical bin packing algorithm to find a (near-)optimal server placement solution.

Thus, RackPacker has three major steps: filtering, bundling, and packing. In the rest of this section, we describe each of these steps in detail.

3.2.3 Filtering and Classification

The goal of filtering is to separate workload trends from noisy transients. A typical approach is to compute a moving average with a sliding window on the power traces, which is equivalent to low-pass filtering. Let S be the set of servers of interest, P_s be the power profile time series of server $s \in S$ with M samples, and T be the sliding window size to compute the moving average. Then, the baseline B_s is computed as $B_s(i) = \frac{1}{T}\sum_{j=(i-T+1)}^{i} P_s(j), i = \{1...M\}$ (with patching zeros when $i \leq T$), and noise $N_s = P_s - B_s$. Figure 3.4 presents the results of filtering the time series shown in Figure 3.2. Figure 3.4(a) is the baseline signal obtained by a 30 minutes moving average. The residual noise signal and its histogram are shown in Figure 3.4(c) and Figure 3.4(d). We use σ_s to represent the standard deviation of the noise.

To obtain and compare the relative times at which different servers peak, we perform discrete Fourier transform (FFT) on the baseline signal. In particular, since the most significant fluctuation has the period of a day, we expect that the second FFT coefficient has the largest magnitude. Indeed, for the power profile in Figure 3.2, the normalized magnitude of the first 10 FFT coefficients are



(a) The baseline trace.

(b) Fourier approximation using 2nd coefficient

(c) The noise trace.

(d) Noise histogram.

Figure 3.4: RackPacker: Filtering and approximation of the power consumption time series.

$[0, 4.2790, 0.2240, 0.7166, 0.4953, 0.1057, 0.1303, 0.0738, 0.0393, 0.0609]$. It is clear that the second component is at least an order of magnitude greater than other components, indicating that it is a good approximation of the overall shape of the power profile.

We denote the second FFT coefficient of the baseline power profile by f_s. Note that f_s is a complex number that represents a sine wave that can be written as $|f_s|Sin(\omega t + \phi_s)$, where $|f_s|$ is the magnitude and ϕ_s is the phase. In a slight abuse of terminology, we call ϕ_s the *primary phase* of the service. For example, Figure 3.4(b) compares the signal

reconstructed by the second Fourier coefficient with the original signal. We clearly see that the second coefficient captures well the overall shape of the original power profile.

Based on the relative magnitudes of the noise level and the fluctuation $|f_s|$, the servers can be classified as *flat* or *fluctuating*. Intuitively, a fluctuating server shows substantial load variation above and beyond its noise. In our example, we consider servers whose power profile has $|f_s| < 3\sigma_s$ to be flat. By this definition, 830 out of the 831 servers fluctuate. Fluctuating servers that show significant phase difference will potentially pack well together, and deserve special attention. This brings us to the bundling step.

3.2.4 Bundling

The goal of bundling is to find small sets of servers whose primary phases "match". Ideally, if the average of f_s across all servers is 0, then the fluctuations cancel each other out . However, in real data centers, this may not be possible. Therefore, the total power load fluctuates at the data center level. Let $\bar{\phi}$ be the average phase of all f_s. Then the best packing approach should spread the data center peak load evenly to all racks. Hence, the target for the bundling process is to make the average phase of each bundle as close to $\bar{\phi}$ as possible.

Another benefit of a common phase for all bundles is dimension reduction. As stated earlier, given a set of power profile time series, we need to verify that at each time instance the total power consumption at each rack does not exceed the power cap with high probability. When server power profiles show distinct phases, we need to perform this verification at the peak time of every power profile. By bundling servers into common phase groups, we only need to verify the time instance when the common phase

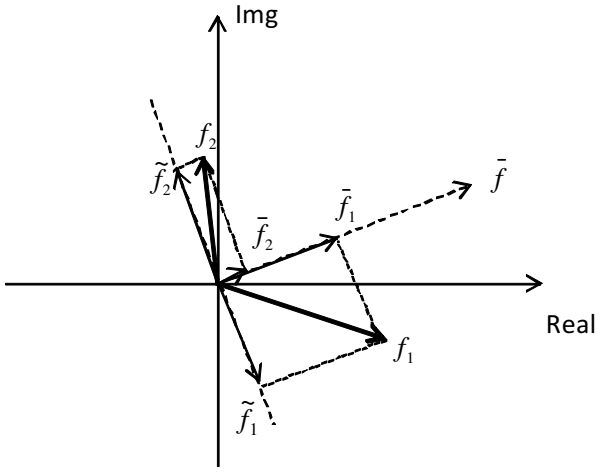


Figure 3.5: RackPacker Bundling: Bundling two servers toward the common phase.

sine wave reaches the peak.

The bundling process can be explained using complex vectors. The complex coefficient f_s of server s can be viewed as a vector in the complex coordinates, as can the average vector $\bar{f}$ with phase $\bar{\phi}$. Then each vector can be decomposed by projecting it to the direction of $\bar{f}$ and to the direction that is orthogonal to $\bar{f}$. Figure 3.5 illustrates this projection. Let f_1 be the 2^{nd} FFT coefficient of server 1, and $\bar{f}$ be the average vector across all servers. Then we project f_1 on $\bar{f}_s$ to obtain $\bar{f}_1$, and then $\tilde{f}_1 = f_1 - \bar{f}_1$. If there exists f_2, whose projection $\tilde{f}_2$ on the direction that is orthogonal to f_s satisfies, $\tilde{f}_2 + \tilde{f}_1 = 0$, then bundling server 1 and server 2 together achieves the common phase. Once common phase bundles are created, further bundling can be performed along the $\bar{f}$ direction so that positive and negative magnitudes cancel each other out .

Algorithm 1 shows the pseudo-code for this bundling step. There are two parameters that affect bundling performance: the max bundle size *BundleCap* and the cancellation threshold ε_B. Intuitively, the smaller we make ε_B, the closer the bundled vectors get to the $\bar{f}_s$ direction. However, one cannot bundle too many servers together since they

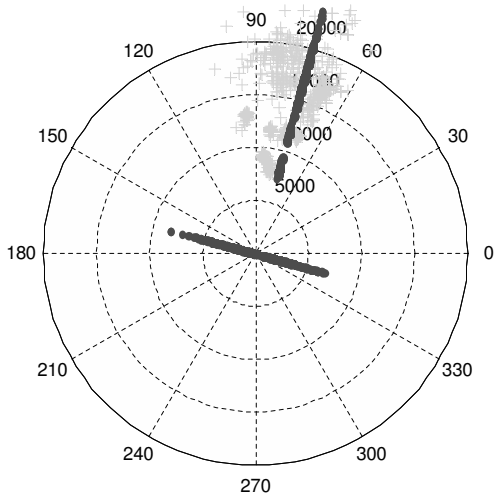
RackPacker: Bundling

1: Compute the mean $\bar{f}$ of $\{f_s\}$ for all fluctuating servers. Compute the angle $\bar{\phi}$ of $\bar{f}$.
2: For each vector f_s with magnitude $|f_s|$ and angle ϕ_s, project f_s to the direction of $\bar{\phi}$ and $\bar{\phi}+\pi/2$:
3: $\quad\bar{f}_s = |f_s| cos(\bar{\phi}-\phi_s)$
4: $\quad\tilde{f}_s = -|f_s| sin(\bar{\phi}-\phi_s)$
5: Sort $\tilde{f}_s$ in a descent order.
6: Select the unbundled server s with the largest $|\tilde{f}_s|$, and place it in a bundle b.
7: Compute the size of $|b|$ and $\tilde{b}$, the length of b along the $\bar{\phi}+\pi/2$ direction.
8: **if** $|\tilde{b}| < \varepsilon_B$ **then**
9: Finish with current bundle and repeat 6.
10: **else**
11: **if** There is no unbundled server **then**
12: Finish.
13: **else**
14: Select unbundled server $s\prime$ such that $|\tilde{f}_{s\prime}+\tilde{f}_b|$ is minimized.
15: **if** the size of $b+s\prime > BundleCap$ **then**
16: Finish current bundle without putting $s\prime$ in b
17: **else**
18: Add $s\prime$ in b, and repeat 7.
19: **end if**
20: **end if**
21: **end if**
22: Treat each bundle as flat. For every bundle b, compute its baseline $B_b = \sum_{s\in b} B_s + \max_{t\in T} |f_b|$, and its variance σ_b from the variance and covariance of the noise vectors of the servers in the bundle.

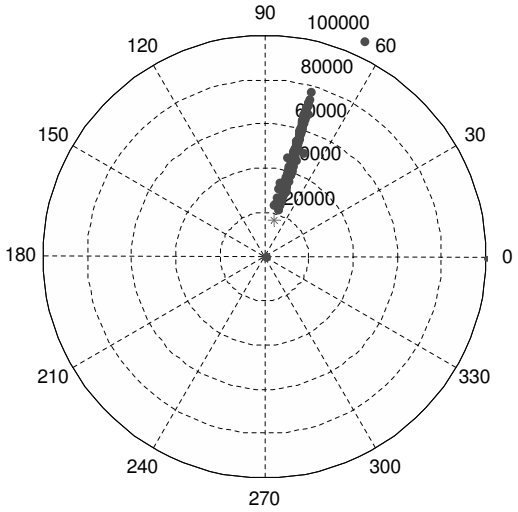
Algorithm 1: Pseudo-code for the Bundling phase of RackPacker

could then exceed the power cap. As we will discuss later, the packing performance is also affected by the correlation of the noise factors. Since noise is not considered in the bundling process, we want to limit the bundling size to give flexibility to the packing step.

Figure 3.6 shows the results of bundling the 830 fluctuating servers in our running example. Figure 3.6(a) shows the original vectors with '+' markers, and their decomposition to the mean and its orthogonal directions with '.' markers. The vectors in the orthogonal directions are canceled out by the bundling process, and Figure 3.6(b) shows



(a) The decomposition of the vectors for 831 servers.



(b) The bundling results of 831 servers based on the decomposition.

Figure 3.6: RackPacker Bundling based on the decomposition of the 2^{nd} FFT coefficient vectors.

the vectors after bundling. The maximum bundle size is 3, when we set the bundle power cap to be one-tenth of the rack power cap.

3.2.5 Packing

Once bundles are created with the same phase, the packing process uses a modified bin packing algorithm for the final placement. A particular challenge that the packing step addresses is the correlations among the spikes.

The goal of the packing phase is to assign bundles to racks in such a manner as to minimize the probability of exceeding the rack power cap. In order to minimize this probability, the packing phase packs together bundles that show minimal correlation in their spikes (noise). Correlated bundles spike in lockstep; this can result in a heightened likelihood of exceeding the rack cap in the event of load spikes such as flash crowds.

In order to compute sets of bundles that show minimal noise correlation, the packing phase proceeds as follows. First, the bundles are ordered in descending order of size. Bundle size for a bundle b is computed as $\sum_{s \in b} B_s + CF * \sigma_b$, where σ_b is the standard deviation of the bundle noise, and CF stands for confidence factor, a configuration parameter (3, here).

We then iterate through this ordered list of bundles and assign them to racks one by one. A bundle b is deemed to fit into a rack r if $\sum_{b' \in r} B_{b'} + B_b + CF * \sigma_{r,b} < C_r$, where $\sigma_{r,b}$ is the standard deviation of the rack noise (= sum of the noise of each bundle in that rack) combined with the noise of bundle b, and C_r is the rack cap. Given a non-empty rack r, to arrive at the next bundle that we'll attempt to pack into r, we order the unassigned bundles in ascending order of their covariance with the current contents

RackPacker: Packing

1: Sort the bundles in descending order by $B_b + CF * \sigma_b$, where CF = confidence factor, a configuration parameter. Call this list L.
2: Pick a bundle b from the top of the list L and assign it to rack R.
3: For all bundles in R, compute $B_R = \sum_{b \in R} B_b$, and $\sigma_R = \sqrt{\sum_{b \in R} \sigma_b^2 + 2 \sum_{b_1, b_2 \in R} covariance(b_1, b_2)}$.
4: **while** list L non-empty **do**
5: Pick a bundle b' from L that is most uncorrelated with all the bundles in R, and add it to R.
6: For all bundles in R, compute B_R and σ_R as above. If $B_R + CF * \sigma_R > C_R$, remove the last bundle from R.
7: **end while**
8: Repeat from 2 with a new rack.

Algorithm 2: Pseudo-code for the Packing phase of RackPacker

Table 3.1: RackPacker Configuration Parameters

Parameter	Value
Rack Cap	11200 W
Bundle Cap	1120 W
ε_B	20
Confidence Factor (CF)	3

of r. We then try to find a bundle from this ordered list that will fit into r. If no such bundle is found, we create a new rack and repeat the process. Algorithm 2 presents the pseudo-code for this phase.

3.3 Evaluation

We have implemented RackPacker in MATLAB. Figure 3.1 shows our choice of parameters for the implementation. The choice of the parameter "Confidence Factor (CF)" is illustrated in figure 3.7. Here assignment confidence is computed as the percentage of racks that fail to stay within the rack cap over a week's trace of data. We see that the

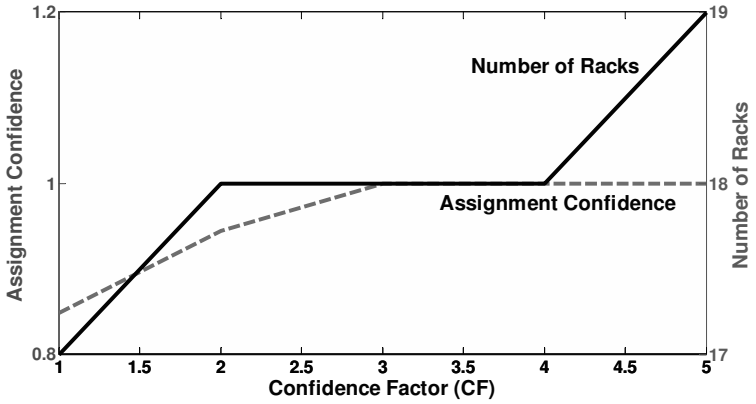


Figure 3.7: Choice of Confidence Factor

choice of the CF value results in a tradeoff between assignment confidence and packing efficiency.

In evaluating RackPacker, we wish to answer the following questions:

1. *How does RackPacker compare with the prevalent server assignment algorithms?* We wish to see if there is a strong argument for using RackPacker in place of existing solutions.
2. *What kinds of workloads is RackPacker best suited for? Conversely, are there workloads for which RackPacker is not suitable?* We wish to know what kinds of applications benefit the most from RackPacker.

We tackle each of these questions in order in this section.

3.3.1 RackPacker: Comparative Performance

To compare the efficacy of RackPacker against current solutions, we use the following metrics:

- **Stranded Power:** This is the difference between provisioned power and actual power consumed per rack. Minimizing stranded resources is the goal of a good provisioning scheme. Hence, the less the stranded power per rack, the better the server assignment algorithm.
- **Packing Efficiency:** This is the number of racks needed to host the given set of servers. We wish to minimize this number in order to improve the utilization of the data center.

Static Assignment Pseudo-code

```
1: Order the servers randomly. Call this list serverlist.
2: Remove the first server s from serverlist and assign it to the first rack. Compute this
   rack's power consumption as: rackpower(1) = power(s)
3: while serverlist is not empty do
4:    Remove server s (of type t, say)from top of serverlist
5:    if Fit Criterion: rackpower(curr_rack)+power(s) < rack power cap then
6:       Assign server s to current rack and update its rackpower
7:    else
8:       Create a new rack, and assign s to it.
9:    end if
10: end while
```

Algorithm 3: Pseudo-code for Static Assignment of servers to racks. Note that power(s) can be the nameplate rating of s, or the peak measured power for s.

We compare RackPacker with two flavors of static assignment: (1) Nameplate Rating-Based assignment, and (2) Peak Power-Based assignment. Both these schemes employ striping, where each type of server is distributed uniformly across all the racks. This results in each rack containing approximately the same relative proportion of each type of server. The *nameplate rating-based scheme* uses the power rating on the server as a measure of its power consumption. Since this number is usually a substantial over-estimate, we also provide a comparison point called the *peak power-based scheme*, which uses the measured peak power consumption of the server in place of the name-

Number of server types		3
Number of servers	Type 1 Type 2 Type 3 Total	329 283 219 831
Average power consumed	Type 1 Type 2 Type 3	199.4 W 194.7 W 210.1 W
Peak power consumed	Type 1 Type 2 Type 3	268.8 W 262.6 W 270 W
Data time-span		1 week

Table 3.2: Description of data using which RackPacker and other solutions are evaluated

plate rating. This is the most aggressive static power provisioning approach, which assumes that the peak in the future does not exceed the peak in the past. Algorithm 3 presents the pseudo-code for both these static assignment schemes. In this section we present analytical results for the nameplate rating-based scheme, and simulated results for the peak power-based scheme and the RackPacker algorithm. In the graphs that we present, the algorithm labelled "Static" refers to the peak power-based scheme.

We evaluate each of these three server assignment algorithms on real power consumption data obtained from a production data center. The data spans 831 servers for a production application. These servers belong to one of three types, corresponding to different tiers of the application. Table 3.2, and figure 3.8 describe the data. The data spans a week, but we train the various algorithms on one day's data, and validate the computed assignment against the remaining days.

Figure 3.9(a) is a pictorial representation of the server assignments computed by RackPacker, and the peak power-based scheme. We find that RackPacker results in 14% more efficient assignment, using only 18 racks against 21 for the peak power-based static assignment. Further, figure 3.9(b) shows the power consumed per rack, averaged over all racks for each of these assignments. The rack cap was assumed to be 11,200 W. We

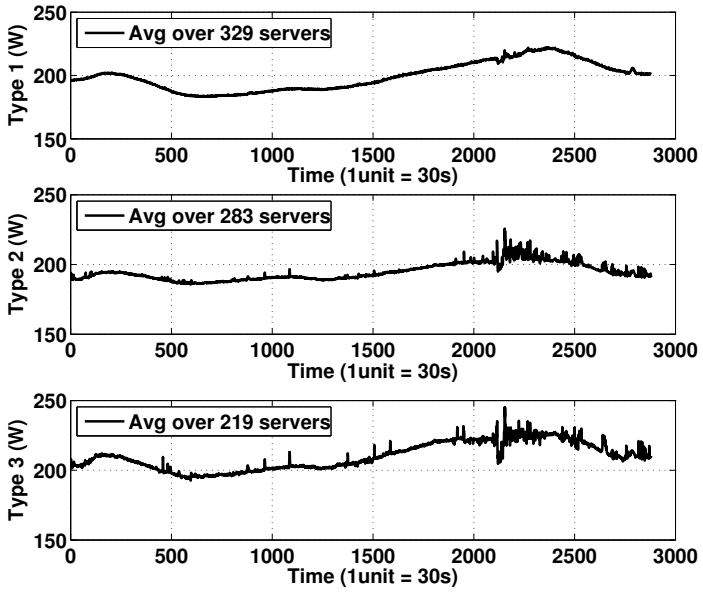
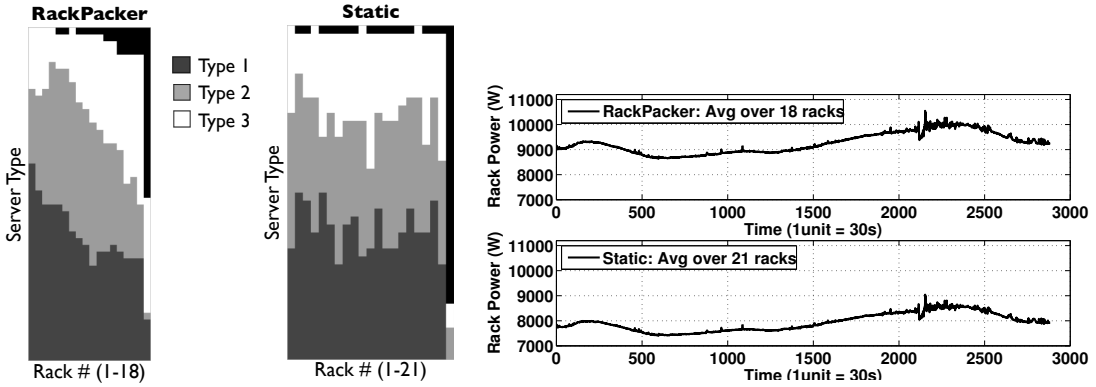


Figure 3.8: Average power consumption behavior for the various server types



(a) Server assignment results (b) Aggregate rack power for computed server assignments

Figure 3.9: Server assignment results for a realistic workload trace.

see that RackPacker results in much less stranded power. RackPacker does much better when compared with the nameplate rating-based scheme. Recall that using nameplate numbers, we need 26 racks to host these servers. Thus here we see a 30% improvement in packing efficiency.

3.3.2 RackPacker: Workload Exploration

In the previous section we showed that RackPacker can improve utilization substantially for a real data center scenario. Now we will explore what kinds of workloads RackPacker is best suited to.

The workload presented in figure 3.8 represents a single-application hosting center. The three types of servers represent three tiers of the application; we see that these tiers operate essentially in lockstep, with load variation being consistent across the tiers. Here we will explore two other data center scenarios. The data for these scenarios is generated through controlled modification of the real data from table 3.2.

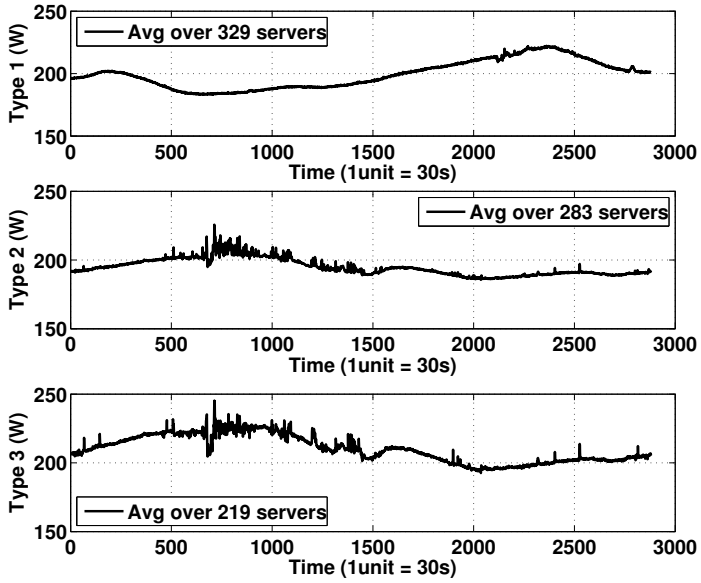
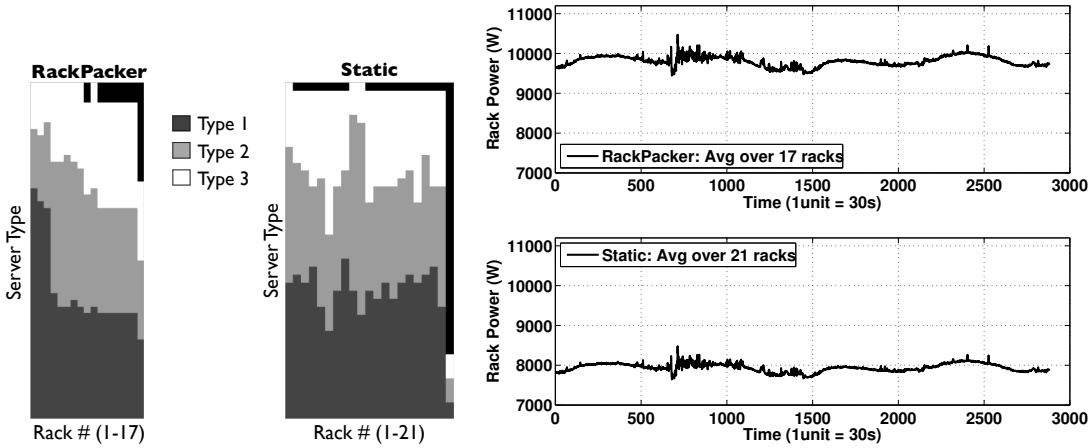


Figure 3.10: Workload with shifted phases: Average power consumption behavior for the various server types

Dedicated Multi-Application Hosting Center: Here we consider data centers that host a small number of applications (more than one). Figure 3.10 shows the data we generated to represent this scenario. Again, there are three types of servers, but types 2 and 3



(a) Server assignment results.

(b) Aggregate rack power for computed server assignment results.

Figure 3.11: Server assignment results for a workload trace with shifted phases.

belong to a different application than type 1—they are thus phase shifted. Figure 3.11(a) shows the server assignment computed by RackPacker and the peak power-based static scheme. Again, we find that RackPacker achieves 19% better packing efficiency, using 17 racks against 21 for the static scheme. Figure 3.11(b) shows the corresponding reduction in stranded power. The nameplate rating-based scheme needs 26 racks (as computed above); RackPacker is now 34% more efficient. In general, we expect that phase shifted servers will benefit more from RackPacker.

Mixed Hosting Center: Here we consider data centers that host a very large number of applications; this represents the cloud computing scenario, where the servers are leased out to various companies that host different applications on them. Figure 3.12 shows the data we generated to represent this scenario. Here we see that there are numerous types of servers, and their correlations are less obvious. Figure 3.13(a) shows the server assignment computed by RackPacker and the peak power-based static scheme. Figure 3.13(b) shows the average rack power utilization for each of these assignments. Again, we find that RackPacker outperforms the static schemes substantially.

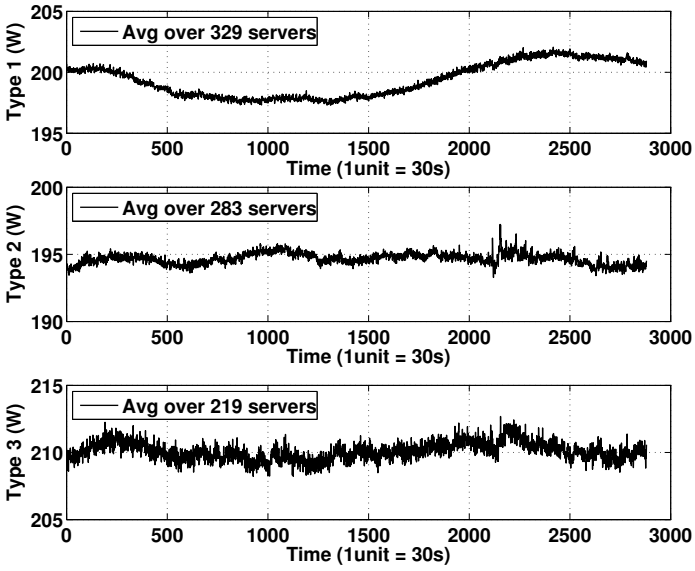
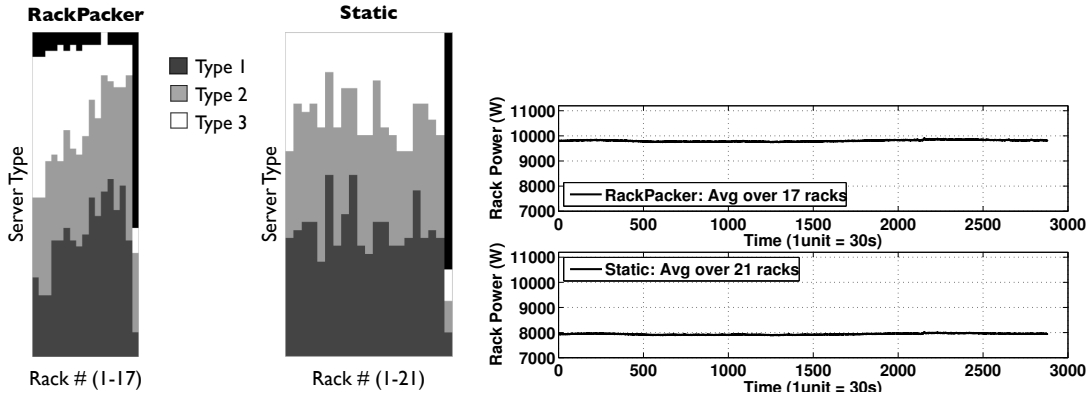


Figure 3.12: Randomized workload: Average power consumption behavior for the various server types



(a) Server assignment results.

(b) Aggregate rack power for computed server assignments.

Figure 3.13: Server assignment results for a workload trace with randomized phases.

3.4 Related Work

In this chapter, we present a scheme for intelligent over-subscription of data center power. The idea of power over-subscription is not new, and has been explored in the

literature in numerous ways. The common theme in prior work, however, is that power tracking and capping are the means used to achieve this over-subscription. To the best of our knowledge, server placement—which sets of servers are placed in which racks—has not been studied as a means of improving data center utilization. Thus, RackPacker is intended to supplement prior work by intelligent server placement that reduces the need for rack-level power capping.

Fan et al. [35] study the aggregate power usage characteristics of large collections of servers for different classes of applications over a period of six months and conclude that cluster-level power capping is a feasible and practical means of improving data center utilization. Their conclusion is based on the intuition that even if power utilization is high at server and rack levels, it is unlikely to be too high at cluster level (since a large number of servers would need to be simultaneously heavily loaded, for this to happen). However, they offer no other insights for implementing power capping.

Muse [25] is a game-theoretic, distributed power management architecture. The goal is to reduce the power consumption of hosted applications by allocating only as many servers as are needed to serve the arriving requests. Muse uses a load prediction model called "flop-flip" which combines two exponentially weighted moving averages of observed load to achieve stable and reasonably agile load estimations. Game theory is used to translate these load estimates to the number of active servers needed per application. Idle servers are shut down to save power.

Chen et al. [28] use two control knobs to restrict application power usage: the number of active servers, as well as their performance states. They use queueing theory to compute request arrival rate over some epoch, and a feedback control loop to correct the predictions over a sub-epoch. Their controller then solves the following optimization problem: given the predicted throughput, what is the optimal number of servers to

allocate for each epoch, and what is the frequency they should each be run at, for each sub-epoch.

Lefurgy et al. [51] use CPU throttle states to implement power capping. CPU throttling reduces the clock speed, with power consumption dropping proportionally. The solution employs a control feedback loop running at each server. The server's power consumption is monitored periodically, and its CPU speed is set to target this load for the next epoch. The authors show how to make this model stable, with bounded settling time.

Heath et al. [45] add a degree of sophistication to their controller by taking into account the heterogeneity of the servers in the data center. Given the bandwidths of all the different resources, the controller's optimization problem is to find the request distribution from clients to servers, and among servers, in such a way that the demand for each resource is not higher than its bandwidth, and we minimize the ratio of cluster-wide power consumption over throughput.

Finally, our idea of translating the server placement problem into a form of multi-dimensional bin packing is inspired by Chekuri et al. [26]. They present an approximate algorithm to pack d-dimensional vectors (servers) into d-dimensional bins (racks) to minimize the maximum load on any dimension. This algorithm, which represents the theoretical best solution for this problem, does not scale well in practice since it requires d to be much less than the average number of servers per rack.

3.5 Summary

Efficient use of data center infrastructure is a pressing issue for the scalability of the IT industry. Due to conservative and static estimation of server power consumption, traditional approaches for power provisioning leave large amounts of provisioned power stranded. RackPacker is a data driven approach for power provisioning. By analyzing real power traces from servers, we obtain the baseline, fluctuation phase, and noise levels for each server. Leveraging this information, we can find sets of anti-correlated servers, in term of both fluctuation phase and noise covariance, that are best candidates for sharing the same rack. Our simulation results from real workload traces show that even with tightly coupled and high utilization services, we can achieve over 30% better packing performance compared to the nameplate rating-based provisioning mechanism. We can save 14% space in comparison to even the most aggressive static assignment approach.

RackPacker works best when there are significant fluctuations on workload and power consumption. There are two reasons that strong fluctuations are increasingly common in server workloads. On-line services are getting more and more geo-focused. That is, many services are designed for users from particular countries or geo-locations. As a result, the workload on these servers reflects usage patterns and the peak load is concentrated in a small time span. Another trend is that the server hardware and software are becoming increasingly power aware. Server idle power is decreasing, while the peak power consumption stays relatively flat. This implies that the power consumption of servers, under variable workload will show fluctuating patterns.

There are several practical concerns when applying RackPacker to real data center operations. We did not consider the rack height constraints when evaluating Rack-

Packer. It is easy to apply rack packing to reduce the power capping if rack height is a constraint. In this case, a data center can add more racks with smaller total power per rack. Sometimes, administrative advantages and security regulations can limit the flexibility of moving services within or across data centers. In addition, current data center networking architecture is hierarchical. Servers are divided into subnets and those in the same rack can only be in the same subnet (VLAN). However, many data centers are dominated by a relatively small number of services each employing a huge number of servers on the same VLAN. Solving the power provisioning problem for these services brings immediate benefits. We did not explicitly address how to proportionally provision cooling with server assignment. Cooling should not be a big concern in this context, since data centers' cooling capacities are designed to match their peak power consumptions.

As a data driven approach for resource management, RackPacker algorithm can be applied to other scenarios, in particular service consolidation via virtualization. Similar to the problem of finding "matching" servers for a rack, one would like to find matching services that can share the same physical server. The difference is that power is an additive resource, ignoring the power factor, but other resources in a physical server may not be additive. For example, depending on cache misses, the time delays of retrieving data from storage can differ significantly when multiple services share the same hardware. Modeling multi-modality resources and optimizing their utilization is challenging future work.

CHAPTER 4

SUPPORT INFRASTRUCTURE: LARGER POWER MANAGEMENT UNITS

No discussion of data center energy management is complete without addressing support infrastructure energy consumption. Data centers are estimated to spend close to 50% of provisioned power on non-IT support equipment such as power delivery and backup, networking, and cooling equipment. Chapters 2 and 3 have discussed ways to reduce idle IT resource energy consumption to achieve power proportionality; however they neglect the significant power overheads from non-IT equipment. This chapter shows how to take a systemic approach to data center power management, achieving power proportionality while also minimizing data center PUE.

4.1 PUE: Where does the power go?

Studies have shown that a state-of-the-art mega data center (housing on the order of 50,000 servers) spends about 59% of the power drawn on IT equipment, 33% on cooling, and the balance 8% on power distribution losses [39]. This translates to a PUE of 1.7 (compare with reported industry average of 2.0 [78]).

An industry rule-of-thumb suggests that for every Watt of energy consumed by a server, about 0.5 W is needed to remove the resulting heat generated [43]. The traditional cooling infrastructure consists of a chiller, a humidifier, and several CRAC (Computer Room Air Conditioner) units. The chiller produces chilled water through refrigeration, and pumps it to the CRAC units. The CRAC units use fans to draw hot air away from the servers and supply them with cool air (using the chilled water from the chiller). The humidifier is used to correct the humidity level of the air leaving the AC units. These are all thermodynamically complex and power-hungry processes.

A significant amount of power (8-10%) is also lost in the power distribution infrastructure [78]. For every Watt of energy used to power servers, up to 0.9 W can be lost in power distribution [76]. To a large extent, these losses result from the series of AC to DC, and DC to AC conversions that are part of the power distribution process. For example, power is typically delivered to a data center as high voltage AC power; this is stepped down to lower voltage AC power for distribution to racks for use by servers and other IT equipment. Inside this IT equipment, power supplies typically convert the AC power to the DC power needed for digital electronics. If the facility uses a UPS, an additional level of indirection is injected in routing the power through the UPS - resulting in another set of AC-to-DC, and DC-to-AC conversions. Power is lost at each of these conversions; further, more power is needed to cool the conversion equipment [76].

In order to prevent outages, data centers use a backup power supply that can kick in temporarily if the primary supply fails. Traditionally, this backup takes the form of a central UPS; power to the facility flows through the UPS, charging it, and is then routed to the racks. Significant power loss can result from this model, as the average UPS has an efficiency of only about 92% [36].

In summary, a considerable fraction of the power consumed by a data center power goes towards non-IT equipment. In fact, industry average energy utilization numbers suggest that almost as much energy goes towards non-IT equipment as is consumed by IT equipment. No data center energy management story can be complete, therefore, without addressing the significant energy consumption of non-IT support infrastructure.

4.2 Related Work

The data center power management space is silo-ed into IT resource management, and support infrastructure management solutions. The former set of solutions has been discussed extensively in chapters 2 and 3. We now survey the solution space for streamlining data center support infrastructure energy consumption. We show that the space is fragmented, with no single systemic approach, and also discuss the significant data center redesign required to deploy these solutions. The next section shows how to address these shortcomings.

The current solution space for reducing support infrastructure energy consumption in data centers consists of point solutions that address individual sources of energy inefficiency. They are engineering techniques targeted specifically at the power distribution, backup, or cooling infrastructure. Accordingly, we categorize them under the following three headings:

1. Power Distribution Efficiency: Power distribution losses result from the multitude of AC to DC and DC to AC conversions that form part of the data center power delivery infrastructure. It has been shown that power conversion efficiency can be improved significantly if the data center is supplied with DC power instead of AC power. DC power delivery systems are up to 20% more efficient that AC delivery [76]. However, moving from AC to DC power delivery can have significant deployment cost.
2. Power Backup Efficiency: The traditional data center power backup solution is a central UPS; significant power loss can result from this model, as the average UPS has an efficiency of only about 92% [36]. A solution to this problem, demonstrated by Google, is to use a distributed power backup model with each server

backed up by its own battery [36]. New facilities can and should use this solution to eliminate power backup losses, but existing facilities face considerable design overhaul if they are to adopt it.

3. Cooling Efficiency: Cooling is one of the most power-hungry processes in a data center, consuming as much as 50% of the power going to the servers. We have described (in chapter 1) a very effective way to reduce this overhead—free cooling. This technique obviates the need for power-hungry chillers by using ambient air to cool the facility. Of course, this is only applicable in facilities with suitable ambient temperature and humidity. Dell recently designed a line of servers that are capable of operating at higher temperatures, thus reducing cooling needs [10]. Another solution is to use a central CRAC controller to better match cooling intensity to facility load and temperature [82]. Our solution can be likened to a distributed controller; one that controls not only CRAC units, but also disks, servers, and power distribution and backup equipment.

4.3 Power-Lean Approach

As we saw in section 4.2, current data center energy management solutions are silo-ed into two separate approaches: power-down solutions for idle IT resources, and engineering solutions for reducing support infrastructure energy consumption. In this section, we show how to shift to a systemic approach, and extend power-down solutions to include support infrastructure. The key insight behind our approach is that, while servers and disks comprise purely IT equipment, racks and larger resource units also include the associated power distribution, backup, and cooling equipment; hence, powering these down will result in a power-proportional solution that also has low PUE. We first formally identify possible units of power management, and then discuss reasons for shifting

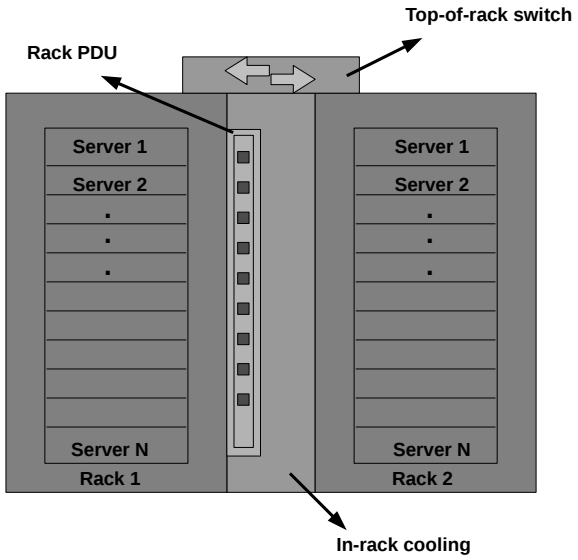


Figure 4.1: Rack Power Cycle Unit

to larger units of power management, and show why it is practical to do so today.

4.3.1 Power Cycle Unit

We define the power cycle unit (PCU) as the resource unit that the power management scheme operates over. This is the unit whose power state is manipulated to track utilization. For example, disk power management schemes manipulate the disk power state (ON/OFF/possibly low-power states corresponding to lower speeds); CPU power management schemes manipulate CPU power (typically through frequency tuning). Our contention is that larger PCU options, which have not been explored thus far, promise significantly bigger energy savings.

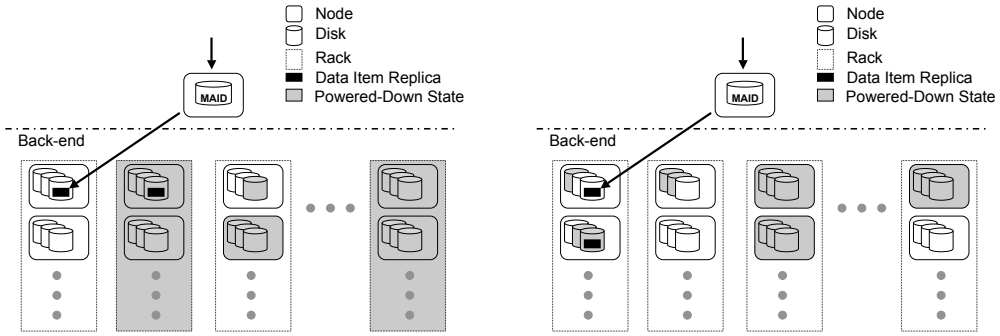
Figure 4.1 illustrates our rack PCU model. Depending on the rack and server dimensions, a rack could contain anywhere between 10 to 80 servers, or more. In figure 4.1

we show a module consisting of two racks, which share an in-rack cooling system, a rack power distribution unit (PDU), and a top-of-rack switch. The in-rack cooling system [14] draws hot air from the servers in the racks, and circulates cool air to maintain the required server operating temperature. This cooling system would typically be allied with a central chiller unit, which would supply it with chilled air; if the outside air conditions are favorable, the chiller can be dispensed with in favor of free cooling. The rack PDU supplies power to the rack components; a switched PDU [20] will allow remote control of this power supply, allowing the rack to be turned on or off over the network. Finally, the top-of-rack switch connects the servers in the rack to the data center network. The switch power is also controlled by the rack PDU. The data center network is typically hierarchical, with rack switches connected using row switches, which in turn connect to a set of central switches that have a link to the outside. In this model, the rack PCU can be powered down/up without impacting the rest of the data center network.

While racks today are physically self-sufficient, and offer fault isolation from the rest of the data center network, powering them down can result in data unavailability or service interruption unless mindful load placement is practised. In order to create rack power-down opportunities, *PCU-aware data organization* must be employed, as follows:

1. Each data item must be spread (striped/mirrored) *across* PCUs, rather than within them. Thus, assuming some degree of data redundancy, one or more host PCUs may be down without impacting the availability of that item.
2. Data access must be localized (as far as possible) to a subset of the PCUs so that others are idle and may be powered down. This is achieved by directing accesses to an item to the more active among its host PCUs.

Figures 4.2(a), and 4.2(b) illustrate PCU = Rack, and PCU = Node, respectively.



(a) PCU=Rack (b) PCU=Node

Figure 4.2: System model

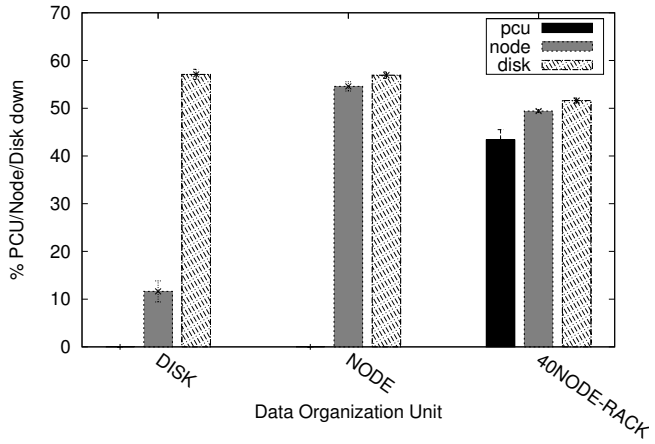


Figure 4.3: Impact of data organization scheme on PCU power-down opportunities

Note how replica placement changes with PCU; note, also, the creation of idle PCUs through selective access of more active replica hosts. Figure 4.3 demonstrates the importance of PCU-aware data organization. We simulate a production data center, and set the PCU to 40-node racks; we then vary the data organization unit (the unit across which replicas are distributed). As expected, we see that unless replicas are distributed across the given PCU (40-node racks, in this case), there are no opportunities for powering them down (number of 40-node PCUs down is zero, when the data organization unit is

disk, or node). When the replicas are distributed across disks, or nodes, we see plenty of disk and node power-down opportunities (number of disks/nodes down is high), but no rack power-down opportunity (number of racks down is zero). Thus, PCU-aware data organization (and retrieval) is key to enabling larger PCUs.

4.3.2 The Case for a Larger PCU

We have presented an overview of the rack PCU model. We make the case that larger PCUs (rack or larger) are needed in order to overcome the limitations of current power proportional solutions. A power management solution that concentrates only on the IT equipment (servers/disks) is limited in its potential benefit due to the nature of the power consumption breakdown of the average data center. Larger PCUs—racks, or containers—allow power cycling of the associated cooling and power distribution equipment as well, thus significantly improving the energy savings potential. We now show that implementing large PCUs is practical today at very little deployment cost; in fact, several current trends among large-scale online services strongly enable this model:

Rack-and-Roll: The online services hosting space is evolving so rapidly that data center design standards are a moving target. However, they are characterized by one guiding principle—modularity. Agility, and rapid scalability are imperatives for successful online services, and both require modularity in design. A need for rapid expansion ushered in the concept of "commodity servers"—pre-assembled servers conforming to the most popular configurations prevalent in industry, ready for purchase off the shelf, deployable simply by plugging them into the data center. The concept has now expanded to racks, which are increasingly becoming the unit of choice for expansion. "Commodity racks" have servers, top-of-rack switches [30], power distribution units [20], and in-rack cool-

ing equipment [14] pre-installed. Purchasing and commissioning a rack is now a mere matter of hours—the "rack-and-roll" phenomenon [29]. Further along this path, entire data centers have now been commoditized—the data center shipping container.

This modularity at multiple levels translates to an opportunity for larger PCUs, as the ability to power down racks, or even entire containers exists today. Each of these potential PCUs houses not only servers and disks, but also their corresponding power distribution, networking, and cooling equipment; powering these down offers energy savings far beyond the limited disk power management space.

Data Model and Placement: Industry-leading storage designs are converging on certain techniques for performance and reliability that prove strongly enabling for power management solutions in general, and large PCUs in particular:

- Replication: Most large-scale systems today replicate their data for fault-tolerance. A replication factor of three is an industry standard [37, 32, 50]. With appropriate replica placement, there is opportunity for powering down one or more replica hosts, without impacting data availability. This provides a tunable parameter—number of live replicas—which can be adjusted based on load, and is a key enabler for storage power management. When combined with PCU-aware replica placement (see trend below), larger PCUs are facilitated.
- Cross-failure-domain replica placement: Each object is replicated, not only across disks, but across racks, and also across data centers. This ensures data availability in the face of domain-correlated failures, such as a rack or data center outage. This practice has been adopted in leading systems like GFS [37], Dynamo [32], and Cassandra [50], among others. Thus, the mechanism is already in place to support PCU-aware data placement.

- Append-only model: A data model that is gaining popularity today due to its performance properties is one where data is stored on disk in immutable data structures. Updates become appends in this model, and consolidation happens lazily. This model caters especially to workloads that are dominated by new writes and large sequential reads, with updates being relatively infrequent. GFS [37], Bigtable [24], and Cassandra [50] are industry-leading systems that use this model. This model is a good fit for power management—updates do not require powering up of all replicas; instead, they can be 'offloaded' (appended) to powered-up disks, and lazily consolidated when the requisite replica hosts are up.

Data and Compute Locality: A challenge in data-intensive compute systems is to localize data and computation. Several techniques have been developed that facilitate this. For example, Bigtable [24] exposes data locality control to its clients, so that they can enforce contiguous placement of related data. Another technique is proposed in GreenHDFS [49], which determines object placement by its age; their measurement of a large Hadoop [2] deployment showed that data popularity is strongly correlated with its age in the system, and by placing data of similar age together, they achieve access locality. Thus, mechanisms are in place today in most production systems to ensure data and compute locality. This facilitates power management, because it allows us to power-manage storage without impacting computation; further, it allows us to power down not just disks, but the associated servers as well—in this model, compute tasks assigned to a server are associated with the data hosted on that server, and thus it is reasonable to infer an idle CPU associated with idle disks.

Data Deluge: Studies suggest that the digital universe—all the digital data being cre-

ated by consumers and businesses worldwide—is growing at a compound annual rate of 57% [53]. Just for the year 2010, this rate of growth translated to an increase in the world's digital data by 1.2 million petabytes [38]. As a reference point, storage capacity growth rate (disk areal density growth) is an estimated 35% [4]—outstripped by data growth. These trends are significantly changing storage needs. Our belief is that we have arrived at a point in the data deluge where the fraction of data accessed, or even accessible, for any reasonable length of time (a week, say), is a tiny fraction of the total data stored. We come, therefore, to the workload property that the vast majority of data is seldom accessed, the data that is accessed is accessed mostly as reads, and writes that are performed are mostly new writes, instead of updates. This property is highly conducive to power management—it creates opportunities for a significant fraction of the storage system to be powered down without impacting performance or data availability.

4.4 Evaluation

Our aim is to quantify the potential energy savings from using larger PCUs, for different data center settings. We wish to answer the following questions:

1. What factors impact optimal choice of PCU?
2. What is the optimal choice of PCU for different data center models, from private to shared, and from brick-and-mortar facilities, to containerized ones?

We describe our methodology, and then present our findings.

4.4.1 Methodology

We use simulations to explore the PCU space, for two reasons: Firstly, for a problem of this scale, a real deployment study is impractical. Secondly, we wish to explore a number of different PCU options, and the large combinatorial space of solutions and their configuration parameters does not allow for a practical deployment study.

Simulator

Our simulator models the power-proportional solution space described in section 4.2, and allows different solutions to be simulated by specifying their architecture and load localization target. The model we work with for our PCU explorations is a MAID-style system, with PCU-aware back-end data organization. Given the system specifications (node and disk capacity, bandwidth, power ratings, PCU membership information, PCU power overhead, and transition time), we simulate the progress of each file request through the system, recording latency and power consumption. Figure 4.2 shows the system model with PCU = Rack, and PCU = Node respectively. Table 4.1 presents the standard simulation parameters.

Data

For our experiments, we use access logs from the Internet Archive's Media Collection [5]. The Internet Archive (IA) makes for a uniquely apt case study in the area of power-aware cloud hosting for a number of reasons: First, it epitomizes the problem of scaling storage to meet the demands of the data deluge—its charter being to store *all* data. Second, the IA targets long-term preservation of (and immediate access to) data, rather than high-throughput data analysis and allied issues; in this it differs from data

Table 4.1: Power-Lean Approach Evaluation: Simulator Parameters (applicable unless specified otherwise)

Parameter	**Description**	**Value**
Data Layout	Redundany scheme employed	PCU-aware, 2-way mirrored
Disk Power (W) (Up/Down/Tran)	Power consumed by disk when up, down, or transitioning between up and down	10/2/10
Node Power (W) (Up/Down/Tran)	Power consumed by node (beyond that consumed by its disks) when up, down, or transitioning between up and down	200/5/200
Rack Power (%) (Up/Down/Tran)	Power consumed by rack (beyond that consumed by its nodes) when up, down, or transitioning between up and down	50/0/50
Disk Access Time (ms)	Time taken to retrieve 1 byte from disk that is up	8
Disk Bandwidth (MBps)	Data transfer rate from disk that is up	100
Disk Transition Time (s)	Time taken by disk to go between up and down states	6
Node Transition Time (s)	Time taken by node (beyond that taken by its disks) to go between up and down states	30
Rack Transition Time (s)	Time taken by rack (beyond that taken by its component nodes) to go between up and down states	300
Power Check Interval (hr)	The intervals at which all PCUs are examined and idle ones powered down	0.5
Power Management Start Time (hr)	The interval after start of simulation when power checking begins	0.5
Disk Power Down Threshold	An exponentially weighted disk access count threshold below which the disk is considered idle	10
Target Disk Down Count	(optional) Force this target number of disks to be powered down during power checks, whether idle or not	50%
Cache Size	MAID disk capacity	100 GB
Number Of Nodes	Number from an IA MC data center	886
Number Of Disks Per Node	Number from an IA MC data center	4

Table 4.2: Power-Lean Approach Evaluation: Trace Characteristics

Attribute	**Trace 1**	**Trace 2**	**Trace 3**
Duration	6 hrs	6 hrs	6 hrs
# accesses	6.5m	7m	6.6m
Avg. access size (MB)	1.7	1.3	1.5
Max access size (GB)	7.73	20.74	7.73
Avg # accesses to a node	7797.77	8338.12	7862.95
Max # accesses to a node	110322	184424	120983
# Nodes accessed	833	838	835

intensive computing services (which have tended to dominate the literature of late—[17, 18, 23]). We believe these are orthogonal problems; once there is a sustainable framework for storing data at truly vast scales, data management/analysis services can be supported in a staged fashion. Finally, the IA is a not-for-profit organization, and operates under constraints (limited resources—money, people) that make the problem of scaling it more challenging; lean operation is not just desirable, it is an imperative in this context.

Table 4.2 gives details of the IA traces we use in our experiments. These traces have a read-ratio ($\frac{\text{\# reads}}{\text{\# accesses}}$) of very close to 1 (0.9926). Unless otherwise specified, each data point presented in the following section is the averaged result of running 6-hour traces from three different days of the week of April 3-9, 2009. (a Monday, Tuesday, and Friday, the same set of hours being picked from each day). The traces are basically HTTP GET logs, and specify, for each file access, the access time, the file details (name, size), as well as the storage node details (id, disk number). However, we manipulate this information slightly to conform to different data organization layouts. Given a data organization scheme—PCU-aware, 2-way mirroring, for example—we statically map each disk to a "mirror disk" such that the mirror disk is on a different PCU from the original disk. An access request to any item on either disk is then directed to the more active of the two. Support for dynamic, per-file mapping is planned in future work.

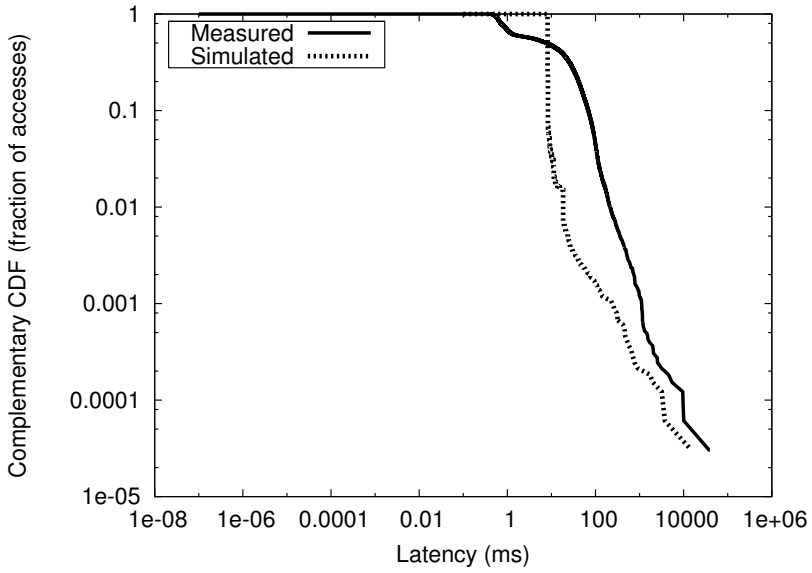


Figure 4.4: Power-Lean Approach Evaluation: Comparing the simulator against Gecko

Validation

We used two methods to ensure that our simulator tracks ground truth. First, we compared its disk-level storage model against measurements from a real storage node. Second, we used actual measurements from production settings to configure the simulator's node- and rack- level parameters.

We used a home-grown low-power storage system called Gecko (see chapter 2: section 2.5) to validate our simulator at disk-granularity. Gecko uses a log-structured storage system layered over RAID-1 block storage. Its log storage model allows it to have control over write accesses—they always go to the disks housing the log head. Its usage of RAID-1 allows it to power down half the disks in the log tail while keeping data live. Our Gecko implementation uses a server with 6 disks, 3 of which are mirrors. In its low-power mode, this implementation keeps the 2 log head disks live, but only 2 out of the 4 disks in the log tail live. Using this low-power mode, we ran our Gecko imple-

mentation on file access traces from 3 of the most-accessed nodes in the IA data. This trace spanned 25 minutes, and comprised 32,749 requests. We also ran this trace on our simulator, configured to resemble the Gecko implementation.

Figure 4.4 compares the measured and simulated access latencies. We do not simulate second-order effects, as we are interested principally in quantifying to a first approximation the energy saving potential of large PCUs. Therefore, the two curves in figure 4.4 diverge in expected ways—the simulator does not model seek distances or seek-optimized request ordering, and as a result does not report latencies below 8 ms (which corresponds to the baseline disk access latency); also, outstanding request queues are not modeled, and so queuing delays are not reported either. That said, we believe the match between the curves is sufficient for our purposes,

We obtained node and rack power cycling information from actual measurements at the IA. These have informed our choice of node and rack transition times, and power overheads.

4.4.2 Results

We now explore the potential as well as the limitations of power management through larger PCUs. We present our results in the context of three motivating scenarios:

Motivating Example: Internet Archive

The Internet Archive's charter is *universal access to all knowledge*. Its knowledge collection currently comprises the Wayback Machine, which stores snapshots of the World Wide Web dating from 1996, and the Media Collection (MC), which stores over 2 PB of

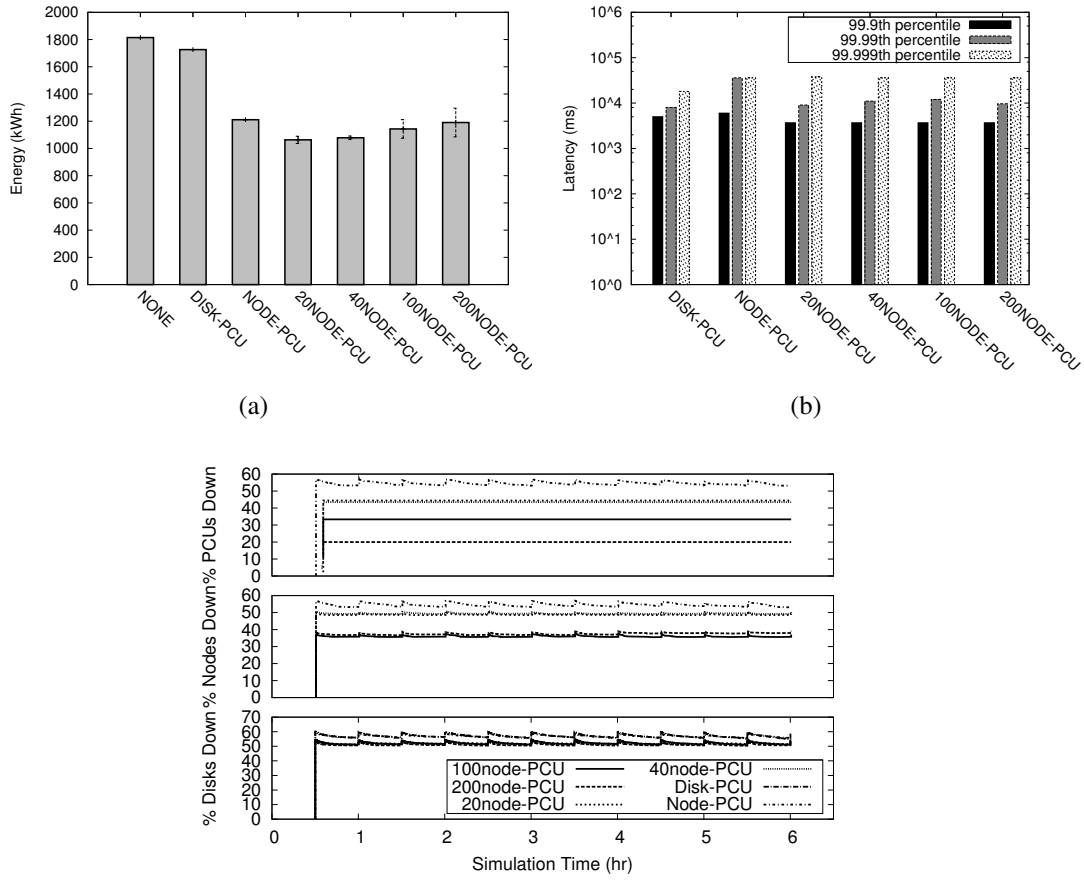
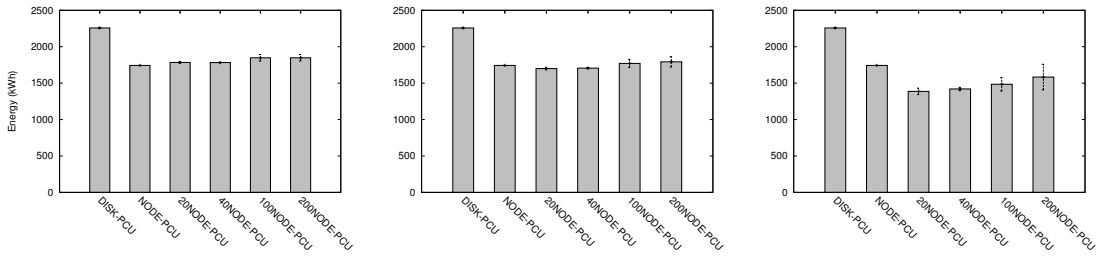


Figure 4.5: Computing optimal PCU size for the Internet Archive

video, audio, image and text files. Our workload data derives from one of the MC data centers; we now explore the right choice of PCU for this data center.

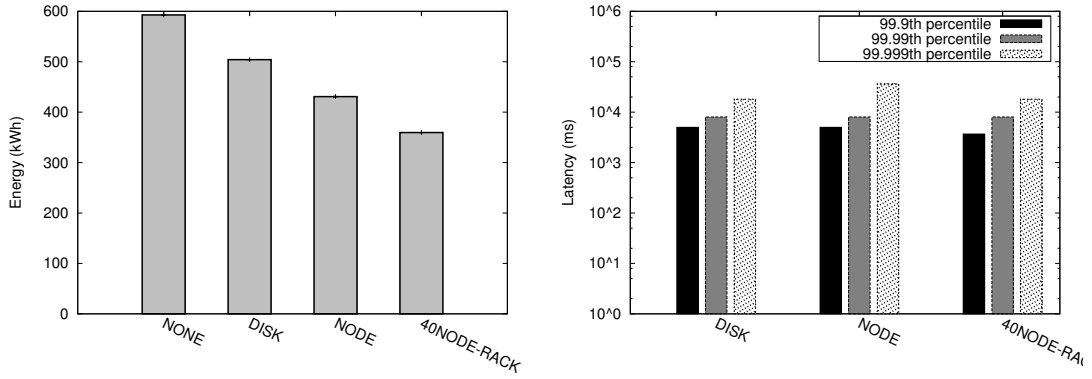
We simulate an MC data center; table 4.1 describes the configuration parameters, which are intended to reflect ground truth. The IA maintains two copies of each file, on two separate storage servers. It reserves two storage servers for new data; when they fill up, two more storage servers are commissioned for the purpose. The MC data center we simulate has 886 storage servers—commodity machines with 4 disks each.

Figure 4.5(a) shows that a 20-, or 40- node rack is the optimal PCU size for this data center, leading to 40% less energy consumption than disk-based power management



(a) PUE=1.05 (b) PUE=1.25 (c) PUE=2.0

Figure 4.6: Effect of PUE on optimal PCU size



(a) (b)

Figure 4.7: Optimal PCU size when disk-to-CPU ratio is 24

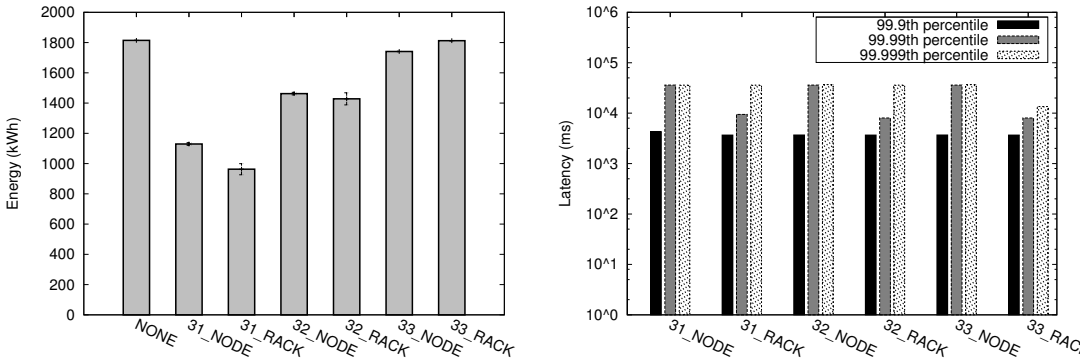
(an 8X improvement!), and 15% less than node-based power management. Further, we see in figure 4.5(b) that the 20- and 40-node PCU configurations outperform the node PCU configuration; each set of three bars in this graph shows the highest latency seen in the 99.9-, 99.99-, and 99.999- th percentile of accesses respectively (left to right). Figure 4.5(c) explains why the rack PCU configurations outperform the node PCU configuration. For each configuration, it tracks the number of PCUs, nodes, and disks that are powered down over the length of the simulation. We see that for all of the configurations with PCU $>$ node, the number of PCUs down stays constant after the initial power check interval. This means that no access goes to a powered-down rack, with the result that rack power-downs have no performance penalty.

Figure 4.6 shows the impact of PUE on optimal PCU size. Rack power overhead reflects data center PUE—50% rack power overhead implies a PUE of at least 1.5. We see that for values of PUE below 1.25, larger PCUs no longer make sense—it is better to use node-based power management in these settings. This bears out our intuition—the motivation for shifting to larger PCUs is to reduce some of the non-IT power overheads of the data center; the smaller these overheads, the less reason to make this shift. Keep in mind, however, that the industry average for data center PUE is 2.

Figure 4.7 shows the impact of disk-to-CPU ratio on optimal PCU size. For a service such as the IA, whose load is entirely I/O-bound, it makes sense to use servers with a larger number of disks. This is in fact precisely the direction the IA is taking; they are in the process of transitioning to storage nodes with 24 to 36 disks each. In this disk-heavy model, we reexamine optimal PCU choice. Figure 4.7 shows that a 40-node rack is still the optimal PCU choice when disk-to-CPU ratio is increased to 24; comparing with figure 4.5 (disk-to-CPU ratio of 4), however, we see that the energy savings over disk- and server-based power management has decreased.

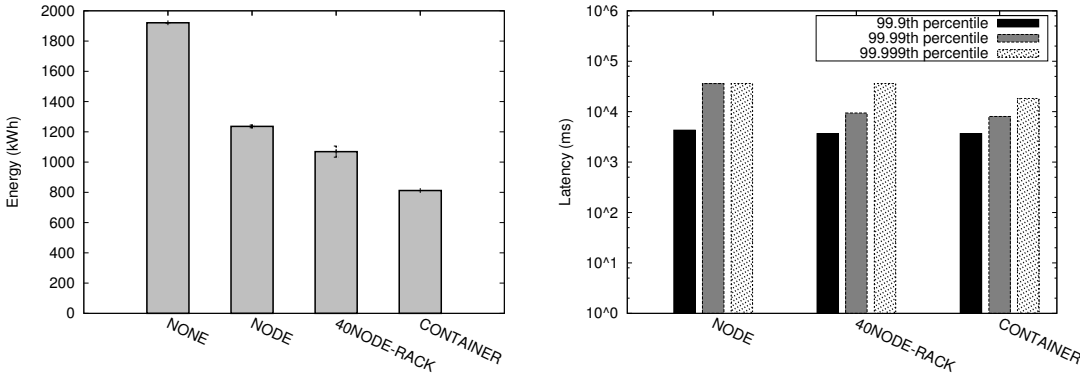
Motivating Example: Amazon S3

We now look at a new online service model that is fast gaining popularity—Storage as a Service (SaaS). Amazon's Simple Storage Service (S3) [1], for example, provides storage at approximately 10 cents per Gigabit-month. The interface is a key-value store. Objects are replicated for reliability - the basic service providing at least 3-way replication, with replicas spread across failure domains such as racks and data centers. Clients can alternatively choose a cheaper option—lower level of replication (Reduced Redundancy Storage (RRS): 2-way, spread across data centers) for data requiring less stringent reliability guarantees.



(a) (b)

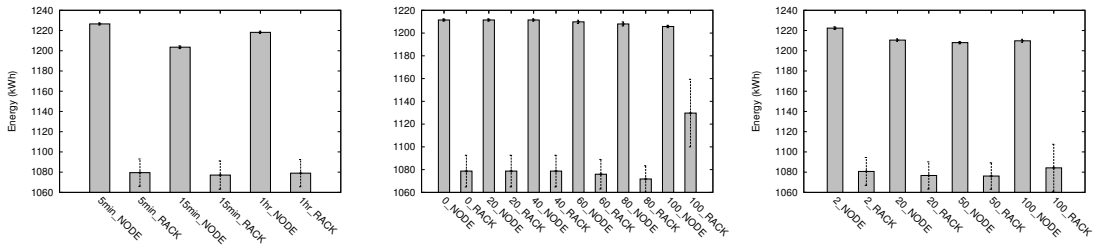
Figure 4.8: Energy savings from tuning number of live replicas



(a) (b)

Figure 4.9: Optimal PCU choice for a container farm

Consider an additional S3 feature: tunable number of live replicas. Clients, when they upload objects, can specify their expected popularity, and tune the number of replicas that need be kept live; the lower this number, the lower the cost of storing the object. With mechanisms already extant for spreading replicas across racks (and data centers), PCU-aware data organization is an easy fit. Figure 4.8 shows the energy savings from reducing the number of live copies. The x-axis labels are of the form rl_PCU, where r is the total number of replicas (3, here), l is the number of live replicas, and PCU can be node or 40-node rack. Keeping only one copy live using rack PCU leads to a 47% energy savings, while keeping two copies live saves 21% energy. Assuming that energy



(a) Power check interval (b) Target disk-down count (c) Disk power-down threshold

Figure 4.10: Result sensitivity to simulator settings

costs contribute 30% to total storage cost, these savings could reduce end-user perceived storage prices by a significant 14%, or 6.3% respectively.

Motivating Example: Container Farm

Containerized data centers are seeing increasing adoption in industry; for example, Microsoft reportedly owns a facility in Chicago comprising 112 containers—a container farm [59]. Containers have the advantages of modularity, ease of deployment, ease of management, and improved space and power efficiency, and might reasonably be expected to be a popular data center commissioning unit of the future. With this in mind, we consider the right PCU choice for a data center consisting of a network of containers.

In this model, we have a new PCU choice—an entire container. The advantage of powering down a container is that we power down its associated power distribution and backup infrastructure. Assuming that these overheads add up to 10% of the power draw, figure 4.9 shows the energy savings from container-based power management. Here we assume that each container has 300 nodes, that container power-up takes 7 minutes (as opposed to 5 minutes for rack power-up), and container power overhead is 60% (as opposed to 50% rack power overhead). We see that the container PCU saves 25% more energy than the formerly optimal 40-rack PCU (figure 4.9(a)), while offering better

performance (figure 4.9(b))! This seeming paradox is explained by the fact that larger PCUs lead to more conservative power management—all the nodes in the container need to be turned off before the container is turned off; as a result, typically only redundant containers get powered down, thus avoiding power-up latency penalties.

Sensitivity Analysis

Finally, we verify that our results are not artifacts of the simulator settings. Figure 4.10 shows that our findings are robust to simulator fine-tuning. Figure 4.10(a) shows that varying the power check interval does not significantly affect the results. In figure 4.10(b) we see that over-aggressive disk power down (forcing 100% of the disks to be down at every power-check interval) can significantly reduce energy savings, but for more reasonable choices of target disk down-count, the results are remain the same. Finally, figure 4.10(c) shows that the choice of optimal PCU size is unaffected by varying disk power-down threshold values.

4.5 Summary

To summarize, we have examined a number of different online service models and shown that in each case significant energy savings are achieved by use of larger PCUs. In the Internet Archive setting, we have shown that using a 40-node rack PCU achieves 8X more energy savings than disk-based power management. This translates to a saving of about 2.6MWh per day over that of a disk-based solution, even for a small 886-node facility. However, we note that the benefit of larger PCUs is strongly tied to the facility PUE—if PUE (and hence rack power overhead) falls below 1.25, larger PCUs are no longer optimal.

We believe that an increasingly likely vision of the future of online services is one where a few infrastructure providers host all of the world's services and data. We show that for an S3-like model, existing data replication and placement policies fit our large PCU model. Further, we show that S3 could provide storage options up to 14% cheaper by adopting rack-based power management, and tuning the number of replicas kept live.

Finally, we examine another point in the design space—container farms. We show that, in this scenario, using entire containers as the PCU leads to an additional energy saving from powering off power distribution and backup equipment (UPS), resulting in a truly power-lean data center.

CHAPTER 5

FUTURE WORK

So far in this dissertation, we have argued the importance of improving data center energy efficiency, and explored various ways to do so. We have identified two main sources of data center energy inefficiency: idle resource energy consumption, and support infrastructure energy consumption. We presented two ways to tackle the former: turn off idle resources (KyotoFS), or maximize resource utilization to reduce resource idleness (RackPacker). To reduce support infrastructure energy consumption, we proposed the use of larger power cycle units. Together these solutions work to streamline data center energy consumption, facilitating sustainable scaling of the cloud computing model.

We now take a step back and examine emerging trends in cloud computing, and discuss some of the challenges facing it.

5.1 Global Network of Data Centers

The cloud computing model shifts data and computing from local servers to a remote platform comprised of servers hosted in a data center somewhere. The closer this data center is to the clients it supports, the better the performance, and the more seamless the aforementioned shift. Therefore, cloud service providers build data centers all around the world, to achieve proximity to as many of their clients as possible. Another reason for data center geo-diversity is failure resilience. With data and computation replicated over data centers spread across the world, service interruption through correlated failures is minimized. Finally, data center geo-diversity can also be beneficial from a cost perspective; certain geographical regions offer ambient conditions that are conducive

to free cooling, while other locations boast cheap land or power. For all these reasons, most major cloud service providers find their operations spread across a global network of data centers, with client requests often spanning multiple data centers. A very relevant question, then, is how to enable efficient inter-data center communication.

We address a subset of this problem in our design of the Smoke and Mirrors File System (SMFS) [80]. SMFS offers efficient, reliable, RTT-independent one-way communication between two data centers. It is intended to achieve file system mirroring across arbitrarily distant data centers with nearly asynchronous performance while offering near-synchronous reliability. SMFS accomplishes this goal through two mechanisms. First, it proactively adds redundancy at the network level to transmitted data. Second, it exposes the level of in-network redundancy added for any sent data via feedback notifications. Proactive redundancy allows for reliable transmission with latency and jitter independent of the length of the link. Feedback makes it possible for a file system (or any other application) to respond to clients as soon as enough recovery data has been transmitted to ensure the desired safety level has been reached. Figure 5.1 illustrates this idea.

Going forward, we would like to explore extending SMFS to enable what we term *energy elasticity* in the cloud. Just as compute and storage elasticity allow clients to control the compute and storage footprint of their applications to match their performance and cost constraints, similarly, energy elasticity would allow them to control their energy footprint. One way for clients to control their energy footprint could be by specifying that applications which do not have real-time constraints be housed on greener facilities, even if they are geographically further away. Extending SMFS to achieve inter data center communication that is nearly independent of inter data center RTT is key to achieving this vision.

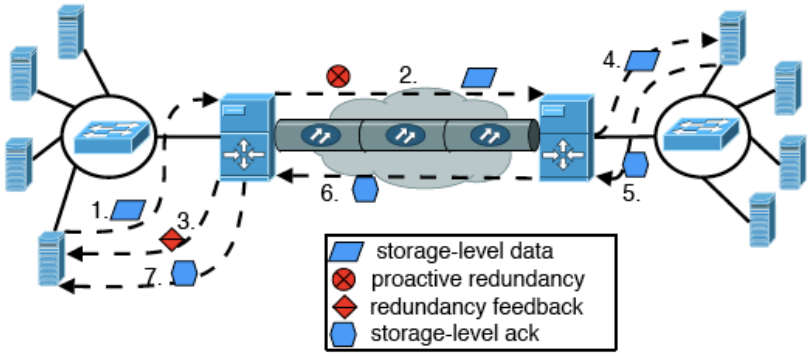


Figure 5.1: The Smoke and Mirrors File System. (1) A primary-site storage system simultaneously applies a request locally and forwards it to the remote mirror. After the network layer (2) routes the request and sends additional error correcting packets, it (3) sends an acknowledgement to the local storage system—at this point, the storage system and application can safely move to the next operation. Later, (4) a remote mirror storage system receives the mirrored request—possibly after the network layer recovered some lost packets. It applies the request to its local storage image, generates a storage level acknowledgement, and (5) sends a response. Finally, (7) when the primary storage system receives the response, it knows with certainty that the request has been mirrored and can garbage collect.

5.2 Cooperative Storage

Hosting data and computation on the cloud can be significantly cheaper and more energy efficient than hosting them locally [82]. The Storage-as-a-Service (SaaS) model has emerged as a direct economic response to growing storage needs, and its proponents argue that an increasingly likely vision of the future is one where a few providers compete to host most of the world's data. Yet enterprises do not find it easy to make the decision to shift their operations to the cloud. This is because there are grave concerns

to trusting one's data to a third party. First, storage providers are fault-prone, leading to data unavailability, or at worst, data loss [7]. Second, data privacy is increasingly a concern, especially when trusting confidential data to a black-box storage provider. Finally, most clients hesitate to get locked down to any one provider and the arbitrary feature set he provides.

We propose Cooperative Storage as an alternate cloud storage model that addresses these concerns. Cooperative Storage creates a unified storage interface from a multitude of storage providers. This model is based on the premise that trusting to a heterogeneity of storage providers is better than trusting to just one (a la the P2P evolution). These storage providers may be derived from SaaS providers as well as private individuals with spare capacity. Data reliability is improved through this cross-domain data spread, fate-sharing is avoided with any one provider, clients are no longer locked down to any single feature set, and finally, capacity can scale to truly global proportions. However, many new concerns emerge, and it is the subject of our current work to address them.

First, designing a distributed storage solution that provides a provably reasonable level of service despite spanning many heterogeneous participants across multiple administrative domains poses several challenges. We are collaborating with subject experts—the authors of the groundbreaking BAR (Byzantine Altruistic Rational) model [16]—to design Cooperative Storage. A BART (BAR-Tolerant) system allows for faulty as well as selfish nodes in a distributed system; it not only tolerates a bounded number of Byzantine nodes (which can deviate arbitrarily from the system protocol), but also an unbounded number of Rational nodes (which deviate from the protocol in ways that increase their net benefit). This model is clearly a good fit for reasoning about the Cooperative Storage solution.

Another, non-technical, concern emerges in the Cooperative Storage model. Content

owners—especially for sensitive data—would hesitate to trust their data to a system where they do not know which entity they can hold accountable for reliably hosting it. The Cooperative Storage model is a form of P2P storage where content hosting responsibility is distributed across a wide network of participants. We are working to incorporate unambiguous responsibility mapping for content in this model.

Finally, a challenge with any large-scale system design, especially when originating in academia, is coming up with a good evaluation plan. We have been collaborating with the Internet Archive (IA) for a few years now, and they are involved in the Cooperative Storage vision. The IA is in the process of venturing into the SaaS space, and is interested in coming up with the right model. We hope to prototype and test our model with the IA.

5.3 Smart Grid

We have already mentioned the immense potential of the cloud to enable a greener way of life. One of the most exciting opportunities it offers is in facilitating a smart power grid. There are pressing economic as well as environmental arguments for the overhaul of the current outdated power grid, and its replacement with a Smart Grid that integrates new kinds of green power generating systems, monitors power use, and adapts consumption to match power costs and system load. Inefficient power generation on the one hand, and severe overload on the other, as well as urgent issues of national and global concern such as power system security and climate change are all driving this shift. As the Smart Grid concept matures, we will see a dramatic growth in green power production: small production devices such as wind turbines and solar panels or solar farms, which have fluctuating capacity outside the control of grid operators. Small com-

panies that specialize in producing power under just certain conditions (price regimes, certain times of day, etc.) will become more and more common. Power consumers are becoming more sophisticated about pricing, shifting consumption from peak periods to off-peak periods; viewed at a global scale, this represents a potentially non-linear feedback behavior. Electric vehicles are likely to become important over the coming decade, at least in dense urban settings, and could shift a substantial new load into the grid, even as they decrease the national demand for petroleum products. The operation of the grid itself will continue to grow in complexity, because the effect of these changing modalities of generation and consumption will be to further fragment the grid into smaller regions, but also to expand the higher level grid of long-distance transmission lines. Clearly, a lot of work is required to transition from the 50-year-old legacy grid of today to the smart grid of the future.

We have worked on identifying some of the computing needs for building this smart grid, and examining the cloud infrastructure to see whether it can address these needs [13]. We show that many promising power management ideas demand scalability of a kind that only cloud computing can offer, but also have additional requirements (real-time, consistency, privacy, security, etc.) that cloud computing would not currently support. Some of these gaps will not soon be filled by the cloud industry, for reasons stemming from underlying economic drivers that have shaped the industry and will continue to do so. However, a focused federal research program could create the needed scalability solutions and then work with the cloud computing industry to transition the needed technologies into standard cloud settings.

As with any new technology, many exciting research questions attend the evolution of cloud computing. We have outlined a few here, with a focus on energy efficiency and sustainability. The process of research is quite as much about finding the right questions, as it is about finding answers. Going forward, we hope to find both.

CHAPTER 6

CONCLUSION

Gartner rates cloud computing as one of the top disruptive technologies of our time [6]. It is not hard to see why: The cloud has the potential to enable everything from ubiquitous computing and universal access to knowledge, to smart power grids, greater social connectivity, and near-infinite extensibility of compute/storage power. It is imperative, therefore, that we work to realize this rich potential and facilitate sustainable evolution of the cloud computing model. This dissertation tackles the question of how to streamline the operation of the global network of data centers that constitutes the cloud, and minimize their energy footprint.

We identify two aspects of data center energy inefficiency: idle resource power consumption, and support infrastructure power consumption. To reduce idle resource power consumption, we can adopt one of two approaches. The first approach tries to match data center power consumption to the load, by turning off resources when they are idle. KyotoFS is an example of this approach, and leverages the log-structured file system to create longer disk idle times, allowing them to be powered down to save energy. The second approach tries to match data center load to provisioned power, by consolidating the load so that resource utilization is improved, and resource idling reduced. RackPacker is an example of this approach; it examines the utilization patterns of servers in the data center to identify near-optimal sets of servers to group together in a rack so that aggregate rack utilization is maximized. Finally, we show how to address support infrastructure power consumption by shifting to larger units of power management in data centers; we argue that turning off entire racks is practical today, and can significantly improve data center energy efficiency.

We also discuss how emerging trends in cloud computing are creating promising avenues for future work. We explore the concept of energy elasticity, which would allow cloud users to control the energy footprint of their applications, in the same way that they control their compute and storage footprint today. We also describe some reliability and security concerns with the current evolution of the cloud storage model, and outline an alternate approach—cooperative storage. Finally, we explore one of the greatest opportunities of cloud computing—enabling the next generation power grid—and discuss the challenges involved. We believe these areas will gain more prominence in the coming years and serve as necessary complements to our work on data center energy management.

The world is moving at an astonishing pace, but it moves in uneven strides. While even household appliances in some parts of the world are connected to the Internet, other parts of the world are yet to see electricity. The cloud has an unparalleled potential to improve technology penetration, and carry the benefits of technology to every part of the world. Already, we see much evidence of this. Telemedicine allows doctors and medical practitioners in urban settings to offer their services remotely to distant rural areas; connected classrooms help teachers leverage the Internet both as a source of information and as a way to connect to a global student base; rural farmers can leverage information kiosks delivering immediate weather and pricing projections—a service critical to their livelihood. The cloud turns infrastructure into utility, and application into service. In doing this, it makes computation suddenly much more accessible. Drinking water, sanitation, electricity, and other utilities reached orders of magnitude more people when they moved from a fragmented to a unified production model; in a similar fashion, the cloud unifies computation and has the potential to bring its benefits to a significantly larger section of the globe. This dissertation is about how to enable responsible evolution of the cloud model, to sustainably realize its rich potential.

GLOSSARY

application a program that runs on one or more computers.

brick-and-mortar data center a data center that is housed in a room or building. See also data center, containerized data center.

cloud computing a model of computing where computation and data reside in servers hosted in remote data centers that are connected by wide-area links. This model turns computation into a pay-per-use utility, similar to electricity, or water. Refer to [56] for a detailed definition. See also server, data center.

computational complexity theory (a quick, informal primer) Problems in class P are those that can be solved in polynomial time, while those in class NP are ones that can be *verified* in polynomial time. Clearly, P$\subset$NP. It is an open question whether P=NP. NP-complete problems are a subset of NP-problems to which any NP-problem can be reduced in polynomial time. NP-hard problems are a set of problems that are at least as hard as NP-complete problems—all NP-complete problems can be reduced in polynomial time to them; however, NP-hard problems need not be in NP.

containerized data center A data center that consists of a shipping container prepopulated with a few thousand servers and associated infrastructure, and can be commissioned and deployed in a matter of months. See also data center, brick-and-mortar data center.

data center A facility dedicated to housing a large group of networked servers and associated power distribution, networking, and cooling equipment, and used to host applications that store, manage, and process digital data. See also server, application.

green data center a data center with a minimal carbon footprint. See also data center.

IaaS Infrastructure-as-a-Service. A cloud computing model where raw compute power is offered as a service. Refer to [56] for a detailed definition. See also cloud computing.

LFS The log-structured file system (LFS) is an append-only file system. It treats the underlying block storage as a log so that all writes become appends, whether they are new writes or updates. Every update, therefore, results in a cascade of updates to all affected meta-data files (inodes) to invalidate old data and update pointers to point to the new data. See [70] for more details.

load balancing Load balancing is a method to distribute workload across multiple computers, network links, disk drives, or other resources, to achieve optimal resource utilization, maximize throughput, minimize response time, and avoid overload.

low-pass filter A filter that passes frequencies below a given value and attenuates frequencies above that value.

MAID Massive Array of Idle Disks. This is a storage model where a set of disks is used as an additional cache layer between main memory and secondary storage (disks). The purpose of this disk cache is to absorb most of the accesses to secondary storage, thus allowing a good fraction of it to be powered down to save energy. Refer to [31] for more detail.

mega data center a data center containing hundreds of thousands of servers or more. See also data center.

nameplate rating The full-load rating of an electrical or electronic apparatus under specified conditions set by the manufacturer.

NP-hard A problem is non-deterministic polynomial-time hard (NP-hard) if solving it in polynomial time would make it possible to solve all problems in class NP in polynomial time. See also computational complexity theory (a quick, informal primer).

online service an application that is available as a service on the Internet, typically accessed with a browser. See also application.

P2P Peer-to-Peer. A decentralized network model, where nodes (peers) are connected together directly rather than via a central server, allowing them to access each other's information directly.

PaaS Platform-as-a-Service. A cloud computing model where virtual machines are offered as a service. Refer to [56] for a detailed definition. See also cloud computing.

personal computing Personal computing encompasses the various uses that individuals put computers to, such as document editing, communication, entertainment.

power capping Power capping mechanisms forcibly curb system power use (by shutting parts of it down, or reducing functionality) when it approaches a specified limit (power capacity, for instance).

power tracking Power tracking is the practice of manipulating system power usage to match its load. See also power-proportionality.

power-proportionality A data center is power-proportional if it uses minimal IT power to execute any given job. Importantly, the data center should consume zero IT power under zero load. See also data center.

private data center A data center that is used to exclusively host one entity's computational needs. Refer to [56] for a detailed definition. See also data center, shared data center.

PUE Power Utilization Efficiency. PUE is defined as the ratio between the total power consumed by a data center, and the power consumed by the IT equipment in the data center. This metric quantifies the amount of power consumed by non-IT equipment such as cooling and power distribution infrastructure, that is not doing *directly* useful work. See also data center.

rack A rack is a frame or enclosure for mounting multiple device modules. Typical server racks can hold a few dozen servers, and are equipped with the required power delivery and network switching gear. See also server.

resource power-down Turning off a device (such as a disk, server, or rack). Powering off a device takes time and consumes energy (during the transition), as does powering it on.

RTT Round Trip Time. A networking term referring to the amount of time it takes a packet to make a round trip from sender to receiver and back.

SaaS Software-as-a-Service. A cloud computing model where applications hosted on cloud platforms are offered as a service. Refer to [56] for a detailed definition. See also cloud computing.

server a computer typically used in enterprise computational settings.

shared data center A data center that hosts more than one entity's computational needs. Refer to [56] for a detailed definition. See also data center, private data center.

SLA Service Level Agreement. This refers to a legal contract that binds a service provider to guarantee specified levels of service to the consumers of the service.

TCO total cost of ownership.

UPS Uninterruptible Power Supply (UPS) is a device that provides battery backup when the electrical power fails or drops to an unacceptable voltage level.

vector bin packing The vector bin packing problem is a multi-dimensional variant of the classical bin packing problem. The latter seeks to find a minimum number of partitions of n real numbers $\in [0,1]$ such that the sum of the numbers in each partition does not exceed 1. The vector bin packing problem, on the other hand, seeks to minimize the number of partitions of n $m-$dimensional vectors $\in [0,1]^d$ such that the sum of each dimension of the vectors in a partition does not exceed 1. See [26] for more details.

virtual machine A virtual machine is a software implementation of a machine that executes programs like a physical machine.

virtualization Virtualization is the creation of a virtual (rather than actual) version of something, such as an operating system, a server, a storage device, or network resources.

BIBLIOGRAPHY

[1] Amazon simple storage service (s3). http://aws.amazon.com/s3.

[2] Apache Hadoop. http://hadoop.apache.org.

[3] HP Power Calculator Utility: A Tool For Estimating Power Requirements For HP ProLiant Rack-Mounted Systems. http://h18004.www1.hp.com/products/solutions/power/index.html.

[4] Moore's law. Wikipedia. http://en.wikipedia.org/wiki/Moore's_law.

[5] The Internet Archive. http://www.archive.org.

[6] Gartner Identifies Top Ten Disruptive Technologies for 2008 to 2012. May 28th 2008. http://www.gartner.com/it/page.jsp?id=681107, Last Accessed: Sep 2011.

[7] From Sidekick to Gmail: A Short History of Cloud Computing Outages. *Network World*, October 12th 2009. http://www.networkworld.com/news/2009/101209-sidekick-cloud-computing-outages-short-history.html, Last Accessed: Sep 2011.

[8] Green Data Center Market to Reach $41 Billion Annually by 2015. Pike Research Press Release, August 5th 2010. http://www.pikeresearch.com/newsroom/green-data-center-market-to-reach-41-billion-annually-by-2015, Last Accessed:Aug 2011.

[9] Impact of Virtualization on Data Center Physical Infrastructure. The Green Grid White Paper, 2010. http://www.thegreengrid.org/~/media/WhitePapers/White_Paper_27_Impact_of_Virtualization_Data_On_Center_Physical_Infrastructure_020210.pdf?lang=en, Last Accessed: Sep 2011.

[10] Dell Leads Shift to "Chiller-Less" Data Centers With Fresh Air Technology. *Dell Press Release*, July 28th 2011. http://content.dell.com/us/en/corp/d/press-releases/2011-07-28-fresh-air-initiative.aspx, Last Accessed: Sep 2011.

[11] Facebook Launches Open Compute Project, 2011. http://www.facebook.com/press/releases.php?p=214173, Last Accessed:Aug 2011.

[12] How Dirty is Your Data? A Look at the Energy Choices that Power Cloud Computing. Greenpeace Report, May 24th 2011. http://www.greenpeace.org/international/en/publications/reports/How-dirty-is-your-data/, Last Accessed:Aug 2011.

[13] Running Smart Grid Control Software on Cloud Computing Architectures. In Workshop on Computational Needs for the Next Generation Electric Grid, April 2011.

[14] 42U. *High Density In-Rack Cooling Solutions for Server Racks, Computer Rooms, Server Rooms & Data Centers*. http://www.42u.com/cooling/in-rack-cooling/in-rack-cooling.htm.

[15] Nitin Agrawal, Vijayan Prabhakaran, Ted Wobber, John Davis, Mark Manasse, and Rina Panigrahy. Design Tradeoffs for SSD Performance. In *USENIX Technical Conference*, June 2008.

[16] Amitanand Aiyer, Lorenzo Alvisi, Allen Clement, Michael Dahlin, Jean-Philippe Martin, and Carl Porth. BAR Fault Tolerance For Cooperative Services. In *ACM Symposium on Operating Systems Principles (SOSP)*, October 2005.

[17] Hrishikesh Amur, James Cipar, Varun Gupta, Gregory Ganger, Michael Kozuch, and Karsten Schwan. Robust and flexible power-proportional storage. In *Symposium on Cloud Computing (SOCC)*, June 2010.

[18] David Andersen, Jason Franklin, Michael Kaminsky, Amar Phanishayee, Lawrence Tan, and Vijay Vasudevan. Fawn: A fast array of wimpy nodes. In *Symposium on Operating Systems Principles (SOSP)*, October 2009.

[19] Thomas Anderson, David Culler, and David Patterson. A Case for Networks of Workstations: NOW. *IEEE Micro*, 15, February 1995.

[20] APC. *Switched Rack PDU*. http://www.apc.com/products/family/index.cfm?id=70.

[21] Luiz Barroso and Urz Holzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 40, December 2007.

[22] E. Carrera, E. Pinheiro, and R. Bianchini. Conserving Disk Energy in Network Servers. In *ACM International Conference on Supercomputing (ICS)*, June 2003.

[23] Adrian Caulfield, Laura Grupp, and Steven Swanson. Gordon: Using flash memory to build fast, power-efficient clusters for data-intensive applications. In *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, March 2009.

[24] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson Hsieh, Deborah Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert Gruber. Bigtable: A distributed storage system for structured data. In *Operating Systems Design and Implementation (OSDI)*, November 2006.

[25] Jeffrey S. Chase, Darrell C. Anderson, Prachi N. Thakar, and Amin M. Vahdat. Managing Energy and Server Resources in Hosting Centers. In *ACM Symposium on Operating Systems Principles (SOSP)*, October 2001.

[26] Chandra Chekuri and Sanjeev Khanna. On Multi-Dimensional Packing Problems. In *Symposium on Discrete Algorithms (SODA)*, January 1999.

[27] Gong Chen, Wenbo He, Jie Liu, Suman Nath, Leonidas Rigas, Lin Xiao, and Feng Zhao. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, April 2008.

[28] Yiyu Chen, Amitayu Das, Wubi Qin, Anand Sivasubramaniam, Qian Wang, and Natarajan Gautam. Managing server energy and operational costs in hosting centers. In *ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, June 2005.

[29] Cisco. *Cisco Data Center Infrastructure 2.5 Design Guide*, 2007. `http://www.cisco.com/en/US/docs/solutions/Enterprise/Data_Center/DC_Infra2_5/DCI_SRND_2_5_book.html`.

[30] Cisco. *Data Center Top-of-Rack Architecture Design*, 2009. `http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9670/white_paper_c11-522337.html`.

[31] D. Colarelli, D. Grunwald, and M. Neufeld. The Case for Massive Arrays of Idle Disks (MAID). In *USENIX Fast and Storage Technologies (FAST)*, January 2002.

[32] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: Amazon's highly available key-value store. In *Symposium on Operating Systems Principles (SOSP)*, October 2007.

[33] Peter Denning. The working set model for program behavior. *Communications of the ACM*, 11(5), May 1968.

[34] Larry Dignan. Google Makes Waves And May Have Solved The Data Center Conundrum. *ZDNet*, September 8th 2008. http://www.zdnet.com/blog/btl/google-makes-waves-and-may-have-solved-the-data-center-conundrum/9937, Last Accessed: Sep 2011.

[35] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power Provisioning for a Warehouse-Sized Computer. In *International Symposium on Computer Architecture (ISCA)*, June 2007.

[36] Katie Fehrenbacher. A Key to Google's Data Center Efficiency: One Backup Battery Per Server. *GigaOm*, April 1st 2009. http://gigaom.com/cleantech/a-key-to-googles-data-center-efficiency-one-backup-battery-per-server/.

[37] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *Symposium on Operating Systems Principles (SOSP)*, October 2003.

[38] Bernard Golden. Cloud computing: How big is big data? idc's answer. *CIO*, May 07 2010. http://www.cio.com/article/593039/Cloud_Computing_How_Big_is_Big_Data_IDC_s_Answer.

[39] Albert G. Greenberg, James R. Hamilton, David A. Maltz, and Parveen Patel. The cost of a cloud: Research problems in data center networks. *Computer Communication Review*, 39, January 2009.

[40] Sudhanva Gurumurthi, Anand Sivasubramaniam, Mahmut Kandemir, and Hubertus Franke. Drpm: Dynamic speed control for power management in server class disks. In *International Symposium on Computing Architecture (ISCA)*, June 2003.

[41] Sudhanva Gurumurthi, Jianyong Zhang, Anand Sivasubramaniam, Mahmut Kandemir, Hubertus Franke, N. Vijaykrishnan, and M. J. Irwin. Interplay of Energy and Performance for Disk Arrays Running Transaction Processing Workloads. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, March 2003.

[42] James Hamilton. On Designing and Deploying Internet-Scale Services. In *USENIX Large Installation Systems Administration (LISA)*, November 2007.

[43] Simon Hancock. Iceland Looks to Serve the World. *BBC Click*, October 9th 2009.

[44] Derrick Harris. What Went Wrong With Iron Mountain's Cloud Storage. *GigaOM*, April 14th 2011. http://gigaom.com/cloud/what-went-wrong-with-iron-mountains-cloud-storage/, Last Accessed:Aug 2011.

[45] Taliver Heath, Bruno Diniz, Enrique V. Carrera, Wagner Meira Jr., and Ricardo Bianchini. Energy Conservation in Heterogeneous Server Clusters. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, June 2005.

[46] David Hornby, Bill Walker, and Ken Pepple. *Consolidation in the Data Center: Simplifying IT Environments to Reduce Total Cost of Ownership*. Pearson Education, 2002.

[47] Aman Kansal, Feng Zhao, Jie Liu, Nupur Kothari, and Arka Bhattacharya. Virtual Machine Power Metering and Monitoring. In *ACM Symposium on Cloud Computing (SOCC)*, June 2010.

[48] James Kaplan, William Forrest, and Noah Kindler. Revolutionizing Data Center Energy Efficiency. *Report by McKinsey&Company*, July 2008. http://www.mckinsey.com/clientservice/bto/pointofview/pdf/Revolutionizing_Data_Center_Efficiency.pdf, Last Accessed:Aug 2011.

[49] Rini Kaushik, Milind Bhandarkar, and Klara Nahrstedt. Evaluation and analysis of greenhdfs: A self-adaptive, energy-conserving variant of the hadoop distributed file system. In *Cloud Computing Technology and Science (CloudCom)*, November 2010.

[50] Avinash Lakshman and Prashant Malik. Cassandra - a decentralized structured storage system. In *ACM SIGOPS Large Scale Distributed Systems and Middleware (LADIS)*, October 2009.

[51] Charles Lefurgy, Xiaorui Wang, and Malcolm Ware. Power Capping: A Prelude to Power Shifting. *Cluster Computer*, 11, June 2008.

[52] Steve Lohr. Data Centers Are Becoming Big Polluters, Study Finds. Oct 16th 2008. http://bits.blog.nytimes.com/2008/05/01/data-centers-are-becoming-big-polluters-study-finds/.

[53] Peter Lyman, Hal Varian, Peter Charles, Nathan Good, Laheem Jordan, and Joyojeet Pal. *How Much Information? Executive Summary*. School of Information Management and Systems, UC-Berkeley, 2003.

[54] Jeanna Matthews, Drew Roselli, Adam Costello, Randy Wang, and Thomas Anderson. Improving the Performance of Log-Structured File Systems with Adaptive Methods. In *ACM Symposium on Operating Systems Principles (SOSP)*, October 1997.

[55] David Meisner, Brian Gold, and Thomas Wenisch. PowerNap: Eliminating Server Idle Power. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, March 2009.

[56] P Mell and Tim Grance. The NIST Definition of Cloud Computing. *National Institute of Standards and Technology (NIST)*, 53(6), 2009.

[57] Robert Metcalfe and David Boggs. Ethernet: Distributed Packet Switching for Local Computer Networks. *Communications of the ACM*, 19, July 1976.

[58] Cade Metz. Google Admits Data Center Podification. *The Register*, April 2 2009. `http://www.theregister.co.uk/2009/04/02/google_data_center_revealed/`, Last Accessed: Aug 2011.

[59] Rich Miller. Microsoft: 300,000 Servers in Container Farm. *Data Center Knowledge*, May 7th 2008. `http://www.datacenterknowledge.com/archives/2008/05/07/microsoft-300000-servers-in-container-farm/`, Last Accessed:Aug 2011.

[60] Rich Miller. Who Has the Most Web Servers? *Data Center Knowledge*, May 14th 2009. `http://www.datacenterknowledge.com/archives/2009/05/14/whos-got-the-most-web-servers/`, Last Accessed:Aug 2011.

[61] Rich Miller. How A Good PUE Can Save 10 MegaWatts. *Data Center Knowledge*, September 13 2010. `http://www.datacenterknowledge.com/archives/2010/09/13/how-a-good-pue-can-save-10-megawatts/`, Last Accessed: Aug 2011.

[62] Justin Moore, Jeff Chase, Parthasarathy Ranganathan, and Ratnesh Sharma. Making scheduling "cool": Temperature-aware workload placement in data centers. In *USENIX Annual Technical Conference*, April 2005.

[63] Ian Murphy. IBM Believes in Commoditised HPC for BI. *ComputerWeekly.com*, June 30th 2010. `http://www.computerweekly.com/blogs/database-notes/2010/06/ibm-believes-in-commoditised-hpc-for-bi.html`, Last Accessed:Aug 2011.

[64] Dushyanth Narayanan, Austin Donnelly, and Antony Rowstron. Write Off-Loading: Practical Power Management for Enterprise Storage. *ACM Transactions on Storage*, 4, November 2008.

[65] Jared Newman. 6 Things You'd Never Guess About Google's Energy Use. *Time Techland*, September 9th 2011. `http://techland.time.com/2011/09/09/6-things-youd-never-guess-about-googles-energy-use`, Last Accessed: Sep 2011.

[66] Oracle. Berkeley db java edition architecture. An Oracle White Paper, September 2006. `http://www.oracle.com/database/berkeley-db/je/index.html`.

[67] Eduardo Pinheiro and Ricardo Bianchini. Energy Conservation Techniques for Disk Array-Based Servers. In *ACM International Conference on Supercomputing (ICS)*, June 2004.

[68] Parthasarathy Ranganathan, Phil Leech, David Irwin, and Jeffrey Chase. Ensemble-Level Power Management for Dense Blade Servers. In *International Symposium on Computer Architecture (ISCA)*, June 2006.

[69] Drew Roselli and Thomas Anderson. Characteristics of File System Workloads. Technical Report No. UCB/CSD-98-1029, 1998.

[70] Mendel Rosenblum and John K. Ousterhout. The Design and Implementation of a Log-Structured File System. *ACM Transactions on Computer Systems (ToCS)*, 10, February 1992.

[71] Gina Smith. The IBM Personal Computer Turns 30. *InformationWeek*, August 1 2011. `http://www.informationweek.com/byte/commentary/personal-tech/desktop-pc/231002983`, Last Accessed: Aug 2011.

[72] Seung Son, Guilin Chen, and Mahmut Kandemir. Disk Layout Optimization for Reducing Energy Consumption. In *ACM International Conference on Supercomputing (ICS)*, June 2005.

[73] Eno Thereska, Austin Donnelly, and Dushyanth Narayanan. Sierra: Practical Power-Proportionality for Data Center Storage. In *Sixth Conference on Computer Systems*, ACM EuroSys, April 2011.

[74] Patrick Thibodeau. Data Centers, Under Strain, Expand At Furious Pace. *Computer World*, May 19th 2011. `http://www.computerworld.com/`

s/article/9216841/Data_centers_under_strain_expand_at_furious_pace_, Last Accessed:Aug 2011.

[75] John Timmer. Datacenter Energy Costs Outpacing Hardware Prices. *Ars Technica*, 2009. http://arstechnica.com/business/news/2009/10/datacenter-costs-outpacing-hardware-prices.ars, Last Accessed:Aug 2011.

[76] My Ton, Brian Fortenbery, and Willian Tschudi. DC Power for Improved Data Center Efficiency. *Report by Lawrence Berkeley National Laboratory (LBNL)*, 2008.

[77] Sandra Upson. Google Watches its Watts. *IEEE Spectrum*, July 2007. http://spectrum.ieee.org/computing/hardware/google-watches-its-watts, Last Accessed: Aug 2011.

[78] Kushagra Vaid. Datacenter Power Efficiency: Separating Fact From Fiction. Invited Talk at USENIX HotPower 2010.

[79] Xiaorui Wang and Ming Chen. Cluster-Level Feedback Power Control for Performance Optimization. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, February 2008.

[80] Hakim Weatherspoon, Lakshmi Ganesh, Tudor Marian, Mahesh Balakrishnan, and Ken Birman. Smoke and Mirrors: Reflecting Files at a Geographically Remote Location Without Loss of Performance. In *USENIX File and Storage Technologies (FAST)*, February 2009.

[81] Aaron Weiss. Can Personal Productivity Live in the Cloud? *Dell*, January 21st 2011. http://content.dell.com/us/en/enterprise/d/large-business/personal-productivity-in-cloud.aspx, Last Accessed:Aug 2011.

[82] Google Whitepaper. Google's Green Data Centers: Network POP Case Study. *Google*, 2011. http://www.google.com/en/us/corporate/datacenter/dc-best-practices-google.pdf, Last Accessed: Sep 2011.

[83] Todd Woody. Google Reveals Its Carbon Footprint. *Forbes*, September 8th 2011. http://www.forbes.com/sites/toddwoody/2011/09/08/google-reveals-its-carbon-footprint, Last Accessed: Sep 2011.

[84] Qingbo Zhu, F. M. David, C. F. Devaraj, Zhenmin Li, Yuanyuan Zhou, and Pei Cao. Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, February 2004.

[85] Qingo Zhu, Zhifeng Chen, Lin Tan, and Yuanyuan Zhou. Hibernator: Helping Disk Arrays Sleep Through the Winter. In *ACM Symposium on Operating Systems Principles (SOSP)*, October 2005.