Towards a Flexible and Energy Adaptive Datacenter Infrastructure

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Muthukumar Murugan

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

David Hung Chang Du

June, 2013

Acknowledgements

There are many people to whom I wish to extend my sincere gratitude for their contribution to my time in graduate school. First and foremost I thank my adviser, guide and mentor Professor David Hung Chang Du. He gave me the opportunity to work on interesting research issues and was nothing but supportive during my entire course of PhD. He motivated me to pursue a PhD degree, encouraged me during my failures, pulled me back to the right path whenever I strayed away and guided me throughout my graduate life. I also thank Dr. Krishna Kant, my co–adviser who introduced me to the research problems that I worked on for this thesis. His constant feedback helped me to get better as I progressed through my PhD degree. I believe that without the invaluable guidance of my two advisers I would not have succeeded in completing this dissertation. I would also like to thank my thesis committee members Professor Jon Weissman, Professor David Lilja and Associate Professor Mohammed Mokbel for their suggestions and feedback. I thank all members of the Center for Research in Intelligent Storage (CRIS) for I was lucky to have worked with them during the course of my PhD.

I would like to specially thank my parents who provided me moral and financial support during the five years of my PhD program. But for their support and encouragement, I would not have contemplated the PhD path. I also extend my gratitude to my uncle Mr. Subramanian and my brother Siva who played a major role in motivating me to pursue a PhD degree. I am eternally indebted to my friends Sureshkumar. N, Pradeep. S, Bhadri Narayan. M, Vignesh. V and Ganesh. K who have continued to stay with me during the joyous and arduous moments of my life.

I would like to thank the anonymous systems staff in the Computer Science Department and Digital Technology Center at the University of Minnesota for their support and help. I thank the reviewers of all the research works that were published during the course of this thesis work for their comments and reviews that helped to shape and guide this dissertation. Last but not the least, I would like to acknowledge the National Science Foundation (NSF) and Center for Research in Intelligent Storage at the University of Minnesota (CRIS–UMN), the sources of funding for the research done during this dissertation.

Dedication

I dedicate this thesis to my family and all my friends for their unconditional love and support and to all my teachers who have helped me to reach this far.

Abstract

The sustainability concerns of Information and Communication Technology (ICT) go well beyond energy efficient computing and require techniques for minimizing environmental impact of ICT infrastructure over its entire life-cycle. The number and popularity of large scale datacenters that host various Internet services, has increased significantly in the recent past. Electricity costs contribute to more than 31% of the overall costs in these datacenters. The increasing energy demand coupled with emerging sustainability concerns requires a re-examination of power/thermal issues in datacenters from a larger perspective of short term energy deficiencies and thermal constraints and ways to make operation of these datacenters more sustainable. The energy deficient scenarios arise for a variety of reasons including variable energy supply and inadequate power, thermal and cooling capacities. Traditionally, ICT infrastructure is overdesigned at all levels from chips to entire datacenters and ecosystem. The paradigm explored in this thesis, called *energy adaptive computing* or *EAC* is to replace overdesign with rightsizing coupled with smarter control. The goal of the Energy Adaptive Computing (EAC) paradigm is to address more directly the issue of sustainability of ICT. This is done by attempting to reduce the carbon footprint of the infrastructure via two mechanisms in addition to intelligent energy management: (a) replacing the wide-spread overdesign of the infrastructure components with rightsizing coupled with smart control to handle occasional overshoot in resource-particularly the energy–requirements, and (b) operation on renewable sources of energy. Renewable energy sources often have variable output and also require intelligent adaptation to the energy envelop.

After a brief introduction to EAC, the challenges associated with EAC in various environments in terms of the adaptation of the workload and the infrastructure to cope with energy and cooling deficiencies, are laid out in detail. This thesis focuses on the issues related to realizing EAC in a cluster environment inside a datacenter. There are three cluster–EAC scenarios studied in detail in this thesis. (1) First, this thesis presents a controller called *Willow* that aims at achieving energy adaptation in a datacenter environment, and addresses the problem of simultaneous energy demand and energy supply regulation at multiple levels from servers to the entire data center. The proposed control scheme adapts the assignments of tasks to servers in a way that can cope with the varying energy limitations. (2) Second, this thesis describes the design and implementation of energy adaptation mechanisms for data centers with potentially multiple tiers of service. Energy adaptation is realized by intelligent allocation of energy at various levels of the hierarchy and shutting down of over-provisioned servers. It is shown that energy adaptation could substantially reduce the power drawn from the conventional electric grid and support most datacenter operations with renewable energy sources and yet provide the required quality of service. This is achieved via coordinated control operations at different time granularities and planning strategies for executing the control operations in order to support different workloads without violating their delay bounds. (3) Finally, this thesis proposes a flexible and energy adaptive object storage framework that can adapt to variations in available or consumable power and its performance is investigated in the context of deduplicated virtual machine disks. The design and implementation of a prototype of the object storage framework is presented. The object storage framework has an adaptive replication mechanism and an adaptive consistency model for the replicas. The replicas of deduplicated virtual machine disks are managed dynamically to provide improved performance and to adapt to power constraints. Smart control techniques are proposed to cope with the power constraints either introduced as a result of increasing node density in the storage arrays or introduced when a mix of renewable (green) and conventional (brown) energy sources are used to power the datacenter. The experimental results demonstrate the ability of the framework to dynamically adapt to the changes in workload and power constraints and minimize adverse performance impacts.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Traditionally, the primary focus of Information and Communication Technology (ICT) research has been the performance improvement of hardware and software. However in the recent past, both cost and environmental concerns have motivated significant research efforts in making the ICT equipment energy–aware and energy–efficient. With the widespread and deepening power/thermal issues, the computing systems are literally becoming power and thermal limited, and a new perspective on computing is required. The power constraints of ICT equipment can arise as a result of both hard caps and soft caps. The hard caps may be because of the capacity constraints of the equipment, for e.g., the maximum power rating of the components. Any violation of hard caps may result in undesirable events like severe damage to the components. Soft caps may arise as a result of optimizations to reduce the overall energy consumption of equipment or to adapt to the load on the devices. Soft caps ensure that there is no load imbalance or degradation of performance in some of the components due to insufficient power availability.

The number and popularity of Internet services ranging from e–commerce (e.g., [1, 2]) to social networking websites (e.g., [3, 4]) have risen significantly in the recent past. Large sever farms host these services and the resources in the server farms are typically accessible over the Internet. Recently, *cloud computing* (here the term *cloud* is a metaphor to the Internet) has gained popularity and is believed to become an important technological solution for meeting future computing needs of both individual users and the enterprise. Cloud computing refers to sharing of resources in large server farms (often called datacenters) by multiple users. The resources are provided as a service to the users. These users can access the resources in the datacenters through the Internet on a pay–per–use basis. This reduces infrastructure investment costs and management costs for the users considerably. On the other hand with the consolidation of a large pool of resources at a single location, power/thermal issues arise at all levels from transistors up to entire ecosystems that involve data centers, clients and the intervening network. The satisfaction of users (e.g., how quickly they can receive the response to a query) of the services that are hosted in these data centers and cost–efficient operation of data centers are important metrics in determining the revenues in these datacenters. Oversizing of datacenters for peak demands is the norm of the day to guarantee *quality of service (QoS)* to users. However this practice is extremely expensive since the additional resources consume significant amounts of energy even when idle. Electricity costs contribute to more than 31% [5] of the overall datacenter costs. Overprovisioning of resources to ensure seamless service during periods of unexpected increase in load (the peak load scenarios that rarely or never happen), leads to unreasonable increase in electricity costs since the overprovisioned resources remain largely underutilized and yet consume significant amounts of power.

Although power/thermal management is an active area of research, power/thermal issues are still largely approached in the form of opportunistic methods to reduce power consumption or stay within the thermal profile while reducing any performance impact. Much of this research in the past has focused on reducing the direct energy usage of the data center, whereas from an environment impact perspective one needs to consider the entire lifecycle of energy consumption – that is, the energy consumption in the manufacture, distribution, installation, operation and disposal of the entire data center infrastructure including IT assets, power distribution equipment, and cooling infrastructure. This has led to efforts in supporting the datacenter operations with renewable energy sources.

Major Information Technology (IT) players like Google and Apple are already running datacenters that use renewable energy sources as much as possible [6, 7]. The energy profiles of renewable energy resources provide ample opportunities for predicting the available energy during different time periods. For instance, the energy available from a solar panel may be correlated with the temperature (e.g., more solar energy during hotter days). The output of solar power plants may also be affected by clouds in the sky and/or the relative position of sun. The available power from wind power plants varies continuously depending on wind strength and directions. Such interactions can be exploited in the adaptation mechanisms. If energy availability is predicted to be low during a certain period of time in a data center located at a specific location, some work can be done during the previous energy plenty periods. For example, in a datacenter supporting a web search application, web crawling and indexing operations can be done during energy plenty periods. Workload can also be migrated from datacenters located in places where there is surplus/cheap energy available. As simple as it may sound, the required control actions are complex and need to be continuously coordinated across multiple time granularities. Infrastructure adaptation in datacenters is not just limited to the use of renewable energy sources. In some developing countries, *brown outs*, which is a reduction in the supply voltage, are common and sometimes are deliberately used as a measure for load reduction [5, 8]. In these cases, it is essential for the infrastructure to adapt to the available power profiles, even in the case of datacenters powered with traditional grid power supply. The alternative to such adaptation mechanisms is to use expensive substitutes like backup generators, UPS etc.

Looking at the datacenter operations from a larger perspective of sustainability concerns and the adaptation techniques to stay within power limitations while still maintaining leaner designs of the infrastructure, this thesis introduces and investigates a new paradigm called *Energy Adaptive Computing* (EAC) [9, 10, 11]. The use of renewable energy sources to power datacenters considerably reduces the environmental impact of these datacenters and has associated challenges. Renewable energy sources are variable and have periods of energy deficiency in their generated power profiles. The main premise of EAC is to provide smarter control in datacenters to facilitate leaner designs and adapt to variations in available or consumable power. The datacenter infrastructure and services need to adapt themselves to variations in the renewable power supply such that the performance degrades gracefully under energy deficient scenarios. An alternative to performance degradation is to increase the proportion of (backup) power drawn from conventional grids (henceforth referred to as brown energy), if available. On the other hand, when the renewable energy available is plentiful, the brown energy consumption can be totally eliminated. This flexibility of operation is important to successfully integrate renewable energy into datacenters.

The goal of this thesis work is to propose infrastructure adaptations in a datacenter from a computer systems perspective so that datacenter operations can be supported (for the most part, if not fully) with renewable energy. The supply constraints introduced in the renewable energy profiles can be substituted (in the presence of stable energy supply) or supplemented by the capacity constraints of the components in the datacenter. This thesis investigates the issues related to adapting to the energy and thermal constraints in the case of different scenarios and workloads. The key difference between typical research works that focus on improving energy efficiency in datacenters and this thesis work is that the primary goal of *EAC* is not to maximize energy efficiency but rather to adapt to the energy and thermal constraints in datacenters. Chapter 2 provides a detailed survey of related research works done in the past on power management in datacenters and in the field of renewable energy powered datacenters. The organization of the rest of this thesis and the contributions of this thesis can be summarized as follows.

1. Introduction and investigation of a paradigm called Energy Adaptive Computing (EAC) in a cluster environment like a large scale datacenter. The objective in EAC is to provide smarter thermal and power management in datacenters and thereby enable downsizing of datacenters and support datacenter operations with renewable energy sources as much as possible. Chapter 3 introduces the new paradigm, energy adaptive computing (EAC) and outlines the challenges associated with realizing EAC in datacenters.

2. The design, implementation and investigation of a control system called *Willow* that does demand–side adaptations in the datacenter via adaptive task migrations in the case of transactional workloads. The proposed system was prototyped on a virtualized cluster and the task migrations were assumed to correspond to virtual machine migrations. Chapter 4 explains the detailed design and implementation of *Willow.*

3. A complete framework to adapt to the power changes in the case of datacenters hosting multi–tiered applications. The multi–tiered (application) datacenter scenario is studied via analytical models and detailed simulations. A key contribution in this piece of work is the planned execution of the control actions that are done to adapt to the variations in energy and workload demands. Chapter 5 explores the specific incarnation of EAC i.e., *EAC in multi–tiered datacenters.*

4. The design and implementation of an energy adaptive object storage framework in a virtualized datacenter. The proposed framework adaptively manages replication and consistency across replicas based on the available power supply and performance requirements of the virtual machines with deduplicated disks. The file system and storage system components are decoupled and a policy engine helps to define the QoS requirements of virtual machines. Chapter 6 details the design factors and implementation details of *flexStore.*

Chapter 2

Survey of Existing Works on Power Management in Datacenters

The power consumption of any device has two components–*static* and *dynamic* [12]. Accordingly the power control techniques that manage these two components are classified as *idle* and *active* power control techniques respectively. Static or idle power consumption is the power consumed by a device even when it is not doing any work. Static power consumption in processors is mainly due to the presence of leakage currents and is a critical issue in modern day processor design. Static power consumed by CPUs is roughly 30% of the peak power consumed. For DRAM it is around 50% and for disks it is around 75% [13]. Dynamic power consumption of a device is due to the circuit activity and is proportional to the frequency of operation of the device. In other words dynamic power consumption is dependent on the utilization of the device. Existing research works on power control in datacenters (e.g., [14, 15, 16]), typically focus on harvesting the idle periods or periods of low activity in the workloads and either put the devices in low power modes or reduce the operation bandwidth of the components. Research works have explored the use of power control techniques in various components and subsystems in the data centers, including CPU [17, 18], memory [12, 19], network links [20, 21] and disks [22, 23]. This chapter presents a survey of the existing research works on power management in datacenters.

2.1 Power Management of (Compute) Servers

Server and cooling subsystems consume around 70% of the overall datacenter power consumption [24]. A number of research works have propopsed techniques to manage the power consumption of servers and make them more energy efficient. These techniques range from architectural changes that make the components more energy efficient to techniques like workload consolidation that migrate the workloads to a fewer servers and enable shutting down the idle servers.

The various circuit level techniques to reduce power consumption, range from optimizations like threshold voltage reduction to adaptive body biasing, transistor sizing, transistor ordering etc. The architectural optimization techniques for power reduction involve reconfiguration of the stages of the pipeline like fetch, decode and memory access. Azizi et al. [25] propose an optimization framework that evaluates energy-performance trade offs at both circuit and architecture levels and builds an integrated design space.

Wang et al. [26] propose an algorithm based on optimal control theory to meet with the energy and thermal constraints in chip multi-processors. Their algorithm exploits the availability of per-core Dynamic Voltage and Frequency Scaling (DVFS) in current day processors and formulates a MIMO model for multi-core processors. Branch prediction techniques try to guess the direction that the instruction flow will take and execute the instructions in the predicted direction. This increases the instruction level parallelism since useful work gets done in the pipeline while waiting for the branch to be resolved. Branch prediction is a significant part in the efficient execution of a program since branches constitute 15–25% of the overall instructions. The accuracy of branch prediction and the complexity involved are crucial factors that influence performance and power consumption. Parikh et al. [18] characterize the power characteristics of different branch predictor algorithms and explore ways to reduce the energy consumption of branch prediction. The authors claim that any increase in the power dissipated by the branch predictor unit would eventually be offset by reduced power consumption in the overall system. Yang and Orailoglu [17] propose a technique to predict the branches earlier so that unnecessary lookups to the Branch Target Buffer (BTB) can be avoided thereby saving considerable power. The authors in [17] introduce a Branch Identification Unit (BIU) that has information about the distance of the next branch instruction and hence avoids unnecessary BTB lookups. The feedback obtained from the compiler, during the runtime can be used find the hot spots within a program. This can be used as a hint to turn off components that are idle. Some of the techniques that are used in such dynamic power control include turning off unused cache lines [12], adjusting the width of the instruction fetch [27] or adjusting the voltage and frequency of operation [28].

Historically, processors have been the largest contributers of power consumption in servers. Due to the research efforts in both industry and academia on processor power consumption, the proportion of power consumed by the processors has been steadily decreasing. Today, memory modules in the servers also consume significant amount of power (40% of the server power consumption [29]). The inherent redundancy and locality in memory accesses leaves room for various optimizations to reduce the energy consumption in the memory hierarchy. Caches are employed in the memory hierarchy to reduce the memory access latency. The performance of caching schemes has a direct impact on the power consumption of memory units since an ideal caching scheme would significantly reduce memory accesses in turn reducing the memory power consumption. K. Kant [19] proposes a control scheme to batch the memory requests in a coordinated manner so that the power consumption of memory modules can be reduced without any degradation of performance. The idea is to dynamically adjust (all or a subset of) the available memory resources for carefully determined time windows so that the degradation in throughput is minimum and the power savings is maximum. Several research works [30, 31, 32, 33] have also proposed techniques to reduce the power consumption of memory buses by modifying the address encoding schemes. Deng et al. [34] propose a scheme to dynamically adjust the active power states (from high power to low power modes) based on the memory bandwidth requirements, energy savings and performance impacts. Their technique takes advantage of the dynamic frequency scaling capabilities of DIMM and adaptively chooses the operational frequency of the memory module depending on the workload demand. Huang at al. [35] propose and implement a power aware virtual memory. Their technique involves tracking and controlling the working set of each process so that the inactive memory components that are least loaded can be transitioned to low power-performance states.

Ranganathan et al. [36] propose to handle the power management of high density blade servers at higher layers (e.g. blade enclosures) rather than at the individual server levels. This coarser grained power control helps to dynamically adapt to the changes in workloads by adjusting the power budgets of the servers in the enclosure. Enclosure level power allocation policies help to ensure that the individual components in the enclosure do not violate their threshold limits. In addition to the changes in architecture and design of datacenters, it is also important to dynamically reconfigure the applications (workloads) and adaptively allocate resources as their demand varies. Moore et al. [37] incorporate temperature profiles in data centers to make workload placement decisions. One of the techniques proposed in [37] is to sort the servers in the order of desirability (e.g., coldest to hottest) and assign workloads to those servers based on the sorted order. Chase et al. [16] propose a system based on economic policies where customers *bid* for resources and energy is also treated as a resource. Resource allocation is done based on the benefits realized from the allocated resources. The system minimizes energy usage while simultaneously dealing with unexpected increases in demand with minimum impact on performance.

Virtualization is increasingly being used in high performance server clusters due to its application isolation capabilities and ease of management. Nagarajan et al. [38] propose a scheme that migrates virtual machines hosting MPI applications from a fault–prone node to a healthy node proactively. Verma et al. [39] investigate the power management in virtualized server clusters hosting HPC applications. They use their experimental results to build a framework for power - aware application placement in virtualized clusters. Their placement strategy takes both CPU usage and working set size into account. Nathuji and Schwan [40] propose a coordinated power management scheme in distributed environments with virtual machines. They leverage the guest level power management techniques and realize them in real time on the host without violating the guaranteed resource isolation between multiple guests. Nathuji et al. [41] leverage their work in [40] and propose a power budgeting scheme that is VM–centric as opposed to typical power budget allocation schemes that allocate power based on the utilization of physical servers. The budget allocation is based on tokens which are assigned to each VM based on the demand and the corresponding power manager modules assign the power budgets relative to the other VMs. X. Wang and Y. Wang [42] propose a cluster level coordinated control architecture that aims at providing per-VM performance guarantees and a cluster level power control.

Chen et al. [43] develop power saving techniques for Internet datacenters that support connection intensive services like instant messaging and online video games where the connections are typically long lived as against the typical short lived request–response web services. They propose schemes that determine the number of servers to be kept powered on and load dispatching algorithms to the servers that reduce the overall energy consumption without sacrificing user experience. This is achieved by means of load skewing that provide opportunities to turn off servers under light loads. Meisner et al. [44] propose a scheme called PowerNap that aims to reduce the idle power consumption in servers by rapidly transitioning to low power states during idle periods and returning to full performance (power) state whenever work arrives so that the work can be completed as fast as possible. Their proposed scheme has minimum impact on latency only when the transition between low power to high power states is small. If the transition latency is high, the request service times are close to the transition latencies especially during low utilization levels when there are a lot of idle periods. Raghavendra et al. [45] propose a coordinated scheme in datacenters that combines different power management solutions at multiple levels in a datacenter (e.g. server, enclosure, rack, datacenter etc.) and ensure that each of these components stay within their power budgets and thermal limitations. The different controllers act at different time windows and coordinate their control actions with the help of feedback mechanisms. The authors also provide a comprehensive analysis of the sensitivity of workloads to the different power management mechanisms.

Barroso and Holzle [13] argue for an energy proportional design of systems in datacenters where the amount of work done by a system is proportional to the energy consumed by it. They observe that current day systems are not energy proportional since the range for dynamic control is very small and a large portion of the energy is static. This is more significantly pronounced especially when the utilization of the system is very low. The authors also claim that improving the dynamic power range of the components and introducing active low power modes where the high–low (and vice versa) transition latency is very small can help to reduce the energy consumption of datacenters significantly.

2.2 Power Management of Networking Subsystem

The power consumed by a network switch developed in [46] is given by Equation 2.1.

$$P_{switch} = P_{chassis} + P_{linecard} \times num_{linecards} + \sum_{i=0}^{i=(N-1)} num\_ports_{config_i} * P_{config_i} \quad (2.1)$$

where $P_{chassis}$ and $P_{linecard}$ are the power consumed by the chassis and the line card respectively. P_{config_i} is the power consumed by a port when it is in configuration i (e.g. 100 Mbps or 1 Gbps).

Power management of networking elements includes both active and idle power control techniques. Idle power control techniques try to minimize the power consumed in switches and links when they are idle by turning them off or putting them to sleep. The active power control techniques adapt the rate of operation of the devices (e.g. network link speed) in order to reduce the operational energy in these devices. Gupta et al. [47] were one of the earliest proponents of techniques to save energy consumption in the network elements. They suggested techniques to slow the clock rates in the network links, put all or few of the subcomponents in switches (e.g. linecards) in sleep modes and reconfiguring the network topologies and one or a combination the above mentioned techniques. Nedevschi et al. [14] investigate both idle and active power management techniques for the network elements. They propose a *"buffer and burst"* scheme that shapes the traffic so that there are alternate active and idle periods and hence provide increased opportunities to sleep. For the dynamic power control of links, the authors propose a scheme where the rate of operation of the network links is dynamically adapted depending on the arrival rate of the packets.

Besides individual power control of the components, research works have also proposed dynamic reconfiguration of networks to save power. Heller et al. [24] propose a dynamic change in the number of active components with changing workload patterns. The goal is to use only a required subset of network components in the datacenter network topology and power down unnecessary components. The proposed system called ElasticTree has a centralized optimizer module that continuously monitors the traffic and computes the optimal network subset and a routing module that redirects the flows as the network capacity increases and decreases dynamically. Mahadevan et al. [48] propose three techniques to conserve energy in datacenter networks that are centrally administrated and provide opportunities for network wide optimizations. The proposed schemes include a strawman approach where the network links are operated at different speeds (e.g. 10 Mbps, 100 Mbps, 1 Gbps etc.). The second technique involves network consolidation where traffic from multiple switches are consolidated and redundant switches are powered down. The third technique involves a coordinated approach where first server consolidation is done by allocating all the jobs to a few servers and then network consolidation is done where redundant network elements are turned off. The authors claim that an energy savings of around 75% can be achieved by combining traffic management and server consolidation techniques.

Ananthanarayanan et al. [49] propose a novel architecture in switches called *"shadow ports"*. Typically when all ports in a switch are in a low power state, all egress packets are buffered and are processed later when the port comes back to the high power state. However ingress packets are lost when a port is in low power state. To overcome this issue, the authors propose a shadow port in each switch which acts as a proxy for the other ports in low power and receives the packets on behalf of them. Agarwal et al. [50]

present a system called Somniloquy that allows operations like background downloads that involve networking elements to continue seamlessly even when the system is put to deep sleep modes. They achieve this with the help of a second embedded processor that is responsible for waking up the system on certain events like packet arrival and keeping the system in active state when required by the applications (like instant messaging, video streaming etc.).

Research works have also investigated energy proportionality of the networking subsystem. Abts et al. [51] propose an energy proportional network topology called the flattened butterfly network that can meet the bandwidth requirements of the datacenter while being energy proportional. The proposed topology is a multi–dimensional direct network and energy proportionality is achieved by exploiting the properties of the topology and the power control mechanisms available in the network links. Mahadevan et al. [52] provide a comprehensive study of configuration and traffic information from real world network traces and propose techniques to make the networking subsystem more power proportional. Kant [20] proposes mechanisms to reduce the power consumption of serial links by means of link width control. Serial links have multiple (4, 12, 16 etc.) *lanes* and the width (bandwidth) of the links is increased by increasing the number of lanes. These links provide the capability to adjust the number of lanes dynamically thereby providing opportunities for dynamic control of the power consumed by these network fabrics. In the proposed link width control algorithm, the number of lanes kept in low power mode is determined based on the link utilization during a recent window.

2.3 Power Management of Storage Subsystem

Gurumurthi et al. [22] propose the use of dynamic speed control of rotating disks based on the load on the disks in order to manage the operational energy consumption of disks. They claim that traditional power control techniques that attempt to spin down disks during idle periods cannot do well in the case of server workloads, since there are not sufficient idle periods in these workloads and predicting the idle period window is extremely challenging. Hence they resort to dynamically adapting the hard disk rotation speeds and hence serve requests at slower speeds thereby reducing the power consumed by the spindle motor. Narayanan et al. [53], on the contrary, claim that there are significant idle periods in the disk workloads of large enterprises and observe that the idle periods are further increased when write offloading is used. Write offloading involves redirecting the write requests to the disks that are spun down to another storage device temporarily and then write the blocks to the original disk later. The spinning down of idle disks combined with write offloading can save upto 60% of energy and there is almost no impact on read/write performance. Zheng et al. [54] propose a storage system called LogStore that uses a two–speed disk. The system uses a disk based log data structure when the utilization demands an increase in speed of operation. This increases the write speed and hence enables the disk to stay at a lower speed longer. Zhu et al. [55] also use multi–speed disks, however the speed change decisions are made at a fairly long (coarse) time granularity (e.g. hours). The long epoch lengths help to amortize the latency cost of speed transitions.

Colarelli et al. [23] propose a system called Massive Array of Idle Disks (MAID). The proposed system uses a massive array of hard disks to replace traditional high performance RAIDs and achieves the same level of performance with a much smaller power budget. The system consists of a few cache disks. As the blocks are accessed, they are copied from non–cache disks to the cache disks and hence the future requests are served from the cache disks and the non–cache disks can remain spun down. Pinheiro et al. [56] propose a similar technique called Popular Data Concentration (PDC) to reduce the power consumption in disk arrays. PDC concentrates the most popular data on a few disks so that the load is skewed and the rest of the disks are idle most of the time. The idle disks can be transitioned to low power modes and since they only contain the least popular data, the impact on performance is minimum. Unlike the MAID technique, where the accessed data is copied (duplicated) to a subset of (cache) disks, in PDC the most popular files are actually migrated to a few disks and there is only one copy of the data. This avoids the increase in space requirements to store the actual amount of data.

Seung Woo Son et al. [57] propose a proactive disk power management scheme where the future idle and active periods are predicted based on hints from the compiler and the appropriate power state for the disks is chosen based on the predicted idle periods. The workloads they investigate are array–based, scientific applications that have specific disk access patterns that can be manipulated in order to improve the locality of disk accesses and introduce the desired idle periods in the disk accesses. Anderson et al. [58] propose a cluster architecture with low-power, cheap processors and flash storage. This architecture performs best in data intensive application scenarios with small sized random accesses.

Amur et al. [59] propose a distributed file system that optimizes its data layout across multiple nodes to achieve power proportional storage. The number of nodes serving the workloads is proportional to the demand. The scaling of nodes is done so that there is no data movement required. The proposed system achieves close to ideal power proportionality even when there are node failures. The proposed system was implemented on the Hadoop Distributed File System (HDFS) [60].

In many cases a coordinated power management between different subsystems in a computing system is often necessary to meet the guaranteed performance requirements. Power management policies that are efficient for a subsystem may result in poor performance of other subsystems. Also any power management policy has to adapt to the workload pattern in order to maintain the same degree of performance. For instance spinning down hard disks might be the most power efficient scheme for a CPU–intensive workload. However when there is a mix of I/O and CPU intensive workloads the additional latency introduced due to the delay in spinning up the disks back may worsen the I/O performance which is already bad. In this scenario, more sophisticated methods need to be designed to maintain a balance between power savings and performance.

2.4 Renewable Energy in Datacenters

Recently research works have explored the use of renewable energy sources to power datacenter operations and the necessary adaptation mechanisms to cope with the variability in green energy supply. Stewart and Shen [61] argue that all the datacenter operations need to be revised in order to use renewable energy to power the datacenters and cope with the intermittent unavailability of power. They suggest a variety of techniques including load balancing across multiple datacenters and re–routing of requests to datacenters that have surplus renewable energy and aggressive prefetching of data from disks before power outage periods. Liu et al. [62] propose to use geographical load balancing in an Internet scale system so that the entire datacenter operations can be managed with significantly small renewable energy capacity and a very small amount of energy storage (e.g. UPS, batteries). They conduct numerical experiments taking into account the spatial and temporal variations in the availability and pricing of renewable energy and the brown (energy from fossil fuels) energy electricity costs. Liu et al. [63] propose a geographical load balancing scheme for Internet–scale systems that is especially beneficial when the electricity is dynamically priced based on the proportion of brown energy used. This dynamic pricing drives the system to rely on green (renewable) energy as much as possible. They also derive optimal solutions for the problem of geographical load balancing where the objective is to minimize the energy cost (number of servers powered on) and the delay cost (loss in revenue). Sharma et al. [5] propose a scheme to handle intermittent energy constraints by applying duty cycles to servers. The servers are turned off and on periodically to reduce their energy consumption. Their technique is a purely power driven management scheme and is independent of workload demands. K. Kant proposes *Energy Adaptive Computing* (EAC) [64] as a solution to the problem of handling variations in energy availability. Kant et al. [10] present a complete design and analysis of a control scheme called *Willow* to achieve QoS guarantees in the presence of variable power supply, as in the case of renewable energy sources. *Willow* considers power and thermal constraints simultaneously and adapts to the variations in workload demand and power supply simultaneously via hierarchical task migration. [9] builds a detailed task model based on QoS requirements of tasks and evaluates *Willow* via more detailed simulations to include the response time as a QoS measure. Murugan et al. [11] investigate the control actions that need to be done in the presence of simultaneous workload and supply variations in datacenters supporting multi–tiered applications. Goiri et al. [65] propose a framework called GreenHadoop which schedules Map–Reduce jobs in datacenters by predicting the availability of green (solar) energy. Their system maximizes the use of renewable energy and tries to minimize the cost of peak brown power usage by scheduling jobs when brown energy is cheap. Brown energy is consumed only when there are violations of time bounds for job completion. [66] also proposes a similar scheduling scheme to maximize green energy usage for parallel batch jobs. Le et al. [67] propose an optimization framework to reduce the brown energy consumption of multi–datacenter services while meeting with the Service Level Agreement (SLA) requirements of these applications. Each datacenter is powered by a mix of green and brown energy and the optimization formulation tries to minimize the overall energy costs by routing the requests to the different datacenters depending on the available green energy. Aksanli et al. [68] propose an adaptive datacenter job scheduler for a mix of batch and interactive jobs that flexibly allocate execution slots to the batch jobs based on the availability of green energy.

Chapter 3

Energy Adaptive Computing and Associated Challenges

3.1 Sustainability and Energy Considerations in Datacenters

A large portion of the power consumed by a data center is either wasted or used for purposes other than computing. In particular, when not managed properly, up to 50% of the data center power may be used for purposes such as chilling plant operation, compressors, air movement (fans), electrical conversion and distribution, and lighting [69]. Furthermore, the operational energy is not the only energy involved here. Many of these functions are quite materials and infrastructure heavy and a substantial amount of energy goes into the construction and maintenance of the cooling and power conversion/distribution infrastructures. In fact, even the "raw" ingredients such as water, industrial metals, and construction materials involve considerable hidden energy footprint in the form of making those ingredients available as usable energy.

It follows that from a sustainability perspective, it is not enough to simply minimize operational energy usage or wastage of components; the energy that goes into the datacenter infrastructure needs to be minimized as well. This principle applies not only to the supporting infrastructure but to the IT devices such as clients and servers themselves. Even for servers in data centers, the increased emphasis on reducing operating energy makes the non-operational part of the energy a larger percentage of the life-cycle energy consumption and could almost account for 50% [70]. For the rapidly proliferating small mobile clients such as cell–phones and PDAs, the energy used in their manufacture, distribution and recycling could be a dominant part of the lifetime energy consumption. Towards this end, it is important to consider data centers that can be operated directly via locally produced renewable energy (wind, solar, geothermal, etc.) with minimal dependence on the power grid or large energy storage systems. Such an approach reduces carbon footprint not only via the use of renewable energy but also by reducing the size and capacity of power storage and power–grid related infrastructure. For example, a lower power draw from the grid would require less heavy–duty power conversion infrastructure and reduce its cost and energy footprint. The down–side of the approach is more variable energy supply and more frequent episodes of inadequate available energy to which the data center needs to adapt dynamically. Although this issue can be addressed via large energy storage capacity, energy storage is currently very expensive and would increase the energy footprint of the infrastructure.

Yet another sustainability issue is the overdesign and over provisioning that is commonly observed at all levels of computer systems. In particular, the power and cooling infrastructure in servers, chassis, racks, and the entire data center is designed for worst-case scenarios which are either rare or do not even occur in realistic environments. Realistic workloads rarely stress more than one resource at a time - for example, it is easy to saturate the CPU when it executes primarily out of the caches, but if a significant memory access activity is involved, CPU will not be able to get the required data quickly enough and will stall. Therefore, taxing CPU and DRAM simultaneously may not even be feasible. It may be possible to saturate both CPU and NIC simultaneously via one or more steady streams of packets of a suitable size, but this may be possible only when incoming packets can be deposited directly into the cache (without a DMA to the memory).

Besides sustainability considerations, a critical factor that impacts the design of datacenters is the power limitations of the infrastructure. For instance racks that used to operate at 8 kW power are now expected to operate at 55 kW due to increased power densities with the adoption of tight enclosures like blade servers [36, 37] in datacenters. The increased power densities in the racks and servers might occasionally cause them to reach their rated power limits and this condition needs to be handled as fast as possible to avoid any adverse impact to the components. The mechanism to handle this might entirely shut down the component in question or migrate workloads away from the power limited servers.

In large data centers, the cooling system not only consumes a substantial percentage of total power (up to 25%) [64] but also requires significant infrastructure in form of chiller plants, compressors, fans, plumbing, etc. Furthermore, chiller plants use a lot of water, much of which simply evaporates. Much of this resource consumption and infrastructure can be done away with by using ambient (or "free") cooling, perhaps supplanted with undersized cooling plants that kick in only when ambient temperature becomes too high. Such an approach requires the energy consumption (and hence the computation) to adapt dynamically to the available cooling ability.

This thesis argues for leaner design of all components having to do with power/ thermal issues: heat sinks, power supplies, fans, voltage regulators, power supply capacitors, power distribution network, Uninterrupted power supply (UPS), air conditioning equipment, etc. This leanness of the infrastructure could be either static (e.g., lower capacity power supplies and heat sinks, smaller disks, DRAM, etc.), or dynamic (e.g., phase shedding power supplies, hardware resources dynamically shared via virtualization). In either case, it is necessary to adapt computations to the limits imposed by power and thermal considerations. In all cases, it is assumed that the design is such that limits are exceeded only occasionally, not routinely.

3.2 Energy Adaptive Computing

Advanced techniques for improving energy efficiency of IT infrastructure and making it more sustainable involve the need to dynamically adapt computation to the suitable energy profile. As mentioned before, in some cases, this energy profile may be dictated by energy (or power) availability, in other cases the limitation may be a result of capacity limitations of the power infrastructure and/or thermal/cooling constraints of the individual components. In many cases, the performance and/or QoS requirements are malleable and can be exploited for energy adaptation. For example, under energy challenged situations, a user may be willing to accept longer response times, lower audio/video quality, less up to date information, and even less accurate results. These aspects have been explored extensively in specific contexts, such as adaptation of mobile clients to intelligently manage battery lifetime [71]. However, complex distributed computing environments provide a variety of opportunities for coordinated adaptation among multiple nodes and at multiple levels [64]. In general, there are three types of distributed energy adaptation scenarios: (a) Cluster computing (or server to server), (b) Client-server, and (c) Peer to Peer (or client to client). These are shown pictorially in Fig. 3.1 using dashed ovals for the included components and are discussed briefly below.

Although in this section, the three scenarios are discussed separately, they generally need to be addressed together because of the complex interactions between them. For example, a data center would typically support both client-server and cluster applications simultaneously. Similarly, a client may be simultaneously involved in both peer-to-peer and client-server applications.

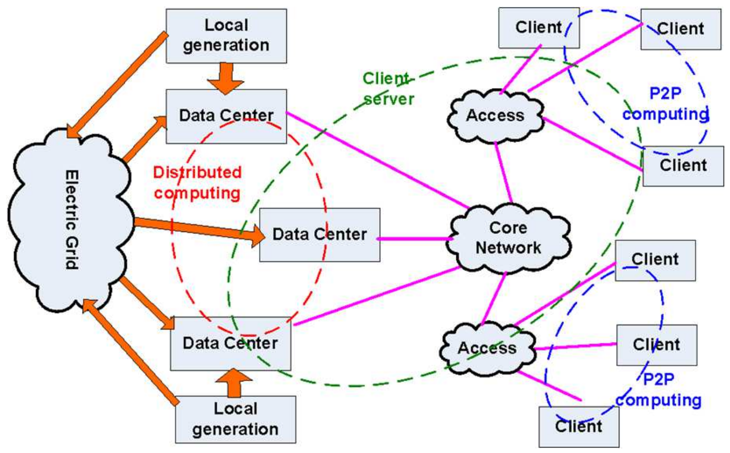


Figure 3.1: Illustration of energy adaptation loops

3.3 Challenges in a Cluster Environment

Cluster EAC refers to computational models where the request submitted by a client requires significant computation involving multiple servers before the response can be returned. That is, client involvement in the service is rather minimal, and the energy adaptation primarily concerns the data center infrastructure. In particular, a significant portion of the power consumed may go into the storage and data center network and they must be considered in adaptation in addition to the servers themselves.

In cluster EAC, the energy adaptation must happen at multiple levels such as subsystems within a server, servers in a rack, etc. At each level there may be a power limit that the level must adapt to. Some of the limits may be "soft" in the sense that they simply represent algorithmic allocation of available energy, and intelligent estimation and adjustment of these limits is crucial. At the highest level, energy adaptation is required to conform to the power generation (or supply) profile of the energy infrastructure. Power limits may also need to be established for different types of infrastructure, for example, the compute, storage and network portions of the infrastructure.

In addition to the available energy, the thermal constraints play a significant role in workload adaptation. Traditionally, CPUs are the only devices that have significant thermal issues to provide both thermal sensors and thermal throttling mechanisms to ensure that the temperature stays within appropriate limits. For example, the T states provided by contemporary CPUs allows introduction of dead cycles periodically in order to let the cores cool. DIMMs are also beginning to be fitted with thermal sensors along with mechanisms to reduce the heat load [72]. With tight enclosures such as blade servers and laptop PCs, ambient cooling, and increasing power consumption, other components (e.g. switching fabrics, interconnects, shared cache, etc.) are also likely to experience thermal issues. In challenging thermal environments, a coordinated thermal management is crucial because the consequences of violating a thermal limit could be quite severe. Also, an over throttling of power to provide a conservative temperature control could have severe performance implications.

Thermal control at the system level is driven by cooling characteristics. For example, it is often observed that all servers in a rack do not receive the same degree of cooling, instead, depending on the location of cooling vents and air movement patterns, certain servers may receive better cooling than others. Most data centers are unlikely to have finer grain mechanisms (e.g., air direction flaps) to even out the cooling effectiveness. Instead, it is much easier to do their thermal management to conform to the cooling profile. So, the simplest scheme is for each server to manage its own thermals based on the prevailing conditions (e.g., on-board temperature measurements). However, such independent controls can lead to unstable or suboptimal control. A coordinated approach such as the one considered in [10] could be used to ensure satisfactory operation while staying within the temperature limits or rather within the power limits dictated by the temperature limit and heat dissipation characteristics.

3.3.1 Estimation and Allocation of Energy

An important aspect of managing a resource is the ability to easily measure resource consumption of the desired software component (e.g., a task, VM, or application) while it is running and accurately estimate the resource requirements before the software is run. The purpose of the latter is to decide where, when, and how to run the software. Unfortunately, when the resource in question is energy (or power), both the measurement or estimation can be quite difficult. Part of the difficulty in measurement arises from the fact that the direct power measurement capability is often unavailable, and the power consumption must be estimated indirectly via available counters in the platform. For example, a direct measurement of power consumption of an individual CPU core is often not feasible and a standard method is to compute power based on a variety of low–level performance monitoring counters that are available on–chip. An even more difficult issue is to break the power consumption of a physical entity down to the software entities (e.g., VMs or tasks) using it [73].

An apriori estimation of power consumption is difficult because the energy consumption not only depends on workload and hardware configuration but also on complex interactions between various hardware and software components and power management actions. For example, energy consumed by the CPU depends on the misses in the cache hierarchy, type of instructions executed, and many other micro-architectural details and how they relate to the workload being executed. Furthermore, when multiple software components are running together on the same hardware, they can interact in complex ways (e.g., cache working set of one task affected by presence of another task). Consequently, neither the performance nor the power consumption adds up linearly, e.g., the active power for two VMs running together on a server does not equal the sum of active powers of individual VMs on the same server. Thus accurate energy estimation remains a difficult problem that we do not tackle in this article.

3.3.2 Planning and Execution of Control Actions

Given the power and QoS constraints, the first step in any power control mechanism is to design an optimal/close–to–optimal solution that would achieve the target performance. However these control actions are not instantaneous and involve overheads. Realizing these solutions in real time involves multiple steps that include time consuming operations like switching a server from sleep/low power modes to fully operational modes or vice versa. The latency involved in these operations is significant and cannot be overlooked. The state changes involved may cause transient instabilities in the participating components. For instance in a datacenter environment, when a server is shut down, the load it was handling needs to be redistributed to other servers. This sudden increase in load in the other servers causes the applications already running on them to slow down. Also, the control actions themselves consume some resources for their execution. All these factors call for a careful planning and execution of the power control actions.

The initiation of the planning process is based on certain events e.g., decrease in available power supply, increase in application traffic etc. If these processes are reactive, i.e., they are initiated after the event has occurred, the associated delays will be extremely large. Hence the events need to be predicted and the necessary control actions need to be initiated beforehand. The planning process can also be made more dynamic by means of Model Predictive Control (MPC) [74] techniques where the actions are planned for every time instant $t + \tau$, $\forall \tau \in \{0, 1, ..T\}$, at time t, and only the control action for time $t + 1$ is implemented. The same process is repeated for the rest of the receding time horizon T. However these techniques could turn out to be expensive when the state space is large. Also these techniques might take a long time to converge and are not essentially optimal.

3.4 Challenges in Other Environments

In this section, we briefly address the client-server and peer-to-peer (P2P) environments. In its full generality, client–server EAC needs to deal with a coordinated end-to-end adaptation including the client, server, and the intervening network. The purpose of the coordination is to optimize the client experience within the energy constraints of each of the 3 components. As the clients become more mobile and demand richer capabilities, the limited battery capacity gets in the way and the energy adaptation can help. Furthermore, since these devices are also constrained in terms of their compute power, energy and network bandwidth, the adaptation goes well beyond just the energy. For example, techniques have been proposed to outsource mobile computation to the cloud platforms that can provide the required resources on demand [75, 76, 77]. In particular, complex and compute intensive image processing tasks can be migrated from the mobile clients to the cloud, and this could be considered as a broader energy adaptation since the migration does conserve battery life. Adding energy adaptation on the server side to these mechanisms makes them particularly difficult to handle. One interesting approach is to adapt the allocation of resources on the server side based on the remaining energy of the mobile clients.

Client–server EAC can be supported by defining client energy states and the QoS that the client is willing to tolerate in different energy states as a contract and then do a contract adaptation, as in [78]. However, since server–side adaptation (such as putting the server in deep sleep state and migrating the application to another server) can affect many clients, the client contracts play a role in where the client applications are hosted on servers and how the servers themselves adapt. This coupling between client and servers, along with appropriate network power management actions makes the overall coordination problem very difficult.

Energy adaptation in P2P environment requires cooperation among peers. This issue is examined in [79]. The authors propose an energy adaptive version of the Bit Torrent protocol. The battery constrained clients define an energy budget for downloading the file which enables them to adapt their contributions to the network and the service they receive from the network based on it. The protocol ensures the provisioning and delivery of the desired service rate to the clients based on their energy budget.

After each transmission or reception, mobile devices continue to remain in active state for a short duration, called tail time, in anticipation of another packet. At high download rates, packets are either received in the tail time or in large single bursts, thus, preventing the frequent occurrence of tail time and reducing the average energy per transfer [80].

3.4.1 Mechanisms to Cope with Energy Limitations

When energy availability is restricted, certain applications – particularly those involved in background activities – don't need to run. Others may run less frequently, with fewer resources, or even change their outputs, and still provide acceptable results. For applications that are driven by client requests and must run, the treatment depends on a variety of factors such as Service Level Agreement (SLA) requirements, level of variability in the workload characteristics, latency tolerance, etc. For example, if the workload can tolerate significant latencies and has rather stable characteristics, the optimal mechanism at the server level is to migrate the entire workload to a smaller set of servers so they can operate without power limitations and shut-down the rest. In this case, a tradeoff is necessary with respect to additional energy savings, SLA requirements, and migration overheads. As the workload becomes more latency sensitive, the latency impact of reconfiguration and power management actions must be taken into account. In particular, if firing up a shut down server would violate latency and response time related SLA, it is no longer possible to completely shut down the servers and instead one of the lower latency sleep modes must be used.

Energy (or equivalently average power over long periods) can be minimized by deliberately increasing power consumption over short periods. For example, it may be possible to minimize energy for a given set of tasks to be executed by running these tasks at the highest speed and then putting the machine in a low–power mode. This race-to-halt policy has traditionally been suboptimal and DVFS (dynamic voltage and frequency scaling) techniques which run components at lower frequencies and voltages so as to raise the device utilization have been more favorable, because of the possibility of significant voltage reductions at low speeds. However, as voltages approach minimum threshold for reliable operation, the voltage differences between various available speeds are becoming rather small. At the same time, the idle power consumption continues to go up due to increased leakage current. In such a situation, race-to-halt becomes an increasingly attractive scheme.

It is important to note that in case of EAC, often the problem is not inadequate work, but rather inadequate energy to process the incoming work. Obviously, in order to reduce the average power consumption, we need to slow down processing, except that this slow–down is not triggered by idling. Unlike the situation where the goal is to minimize wasted energy, an energy constrained environment requires careful simultaneous management of multiple subsystems in order to make the best use of the available energy. For example, it is necessary to simultaneously power manage CPU, memory and IO adapters of a server in order to ensure that the energy can be delivered where most required. The admission control, migration and power management of a large number of resources at multiple levels raises a lot of interesting issues in terms of the stability and optimality of the control in addition to the issues of the overhead and lag associated with information exchange. A comprehensive control theoretic framework is required in order to address these issues [26, 42]. When the control extends over multiple physical facilities, perhaps each with differing energy costs, the problem becomes even more complex.

Although much of the above discussion concerns servers, similar issues apply to clients and their subsystems. For example, the partitioning and control of power between CPU, memory, storage and other portions of a client involves the same set of issues as servers. However, the peer-to-peer interaction between clients involves some unique issues as already stated above.

Storage and network also need serious consideration in energy adaptive computing because of increasing data intensiveness of most applications. Energy management of rotating magnetic media often involves long latencies (in spinning down or spinning up the drives) and reliability issues resulting from RPM changes or repeated starts and stops. The emerging solid state storage (SSD) can be helpful in this regard. The energy management of network devices such as switches and routers is inherently difficult because of its nonlocal impact. For example, if a router/switch port is placed in a low power mode, every application and endpoint whose traffic goes through this port will be affected.

While the topic of applications changing their behavior in the face of energy limitations has been explored in several specific contexts such as audio/video streaming, rendering a web page, P2P content sharing [71, 81], and mechanisms to specify and manage the adaptation have been proposed [78, 82], it is apparent that there is scope for considerable further work on how and when to apply various kinds of adaptation mechanisms (e.g., lower resolution, higher latency, control over staleness and/or accuracy, etc.) under various kinds of power/thermal limitation scenarios. The main theme in EAC has been to cut down "fat" at all levels and thereby lower not only the direct energy consumption but also the entire life-cycle energy costs that are essential to examine from a sustainability perspective. This leanness has a down-side: the increased fragility in the system which can be exploited by attackers. In particular, just as current systems can be victimized by denial of service (DoS) attacks, the systems proposed here can be further victimized by denial of energy (DoE) attacks. For example, it is possible to craft "power viruses" whose aim is to consume as much power as possible. A carefully planned attack using such viruses can significantly disrupt a distributed EAC scheme and lead to instabilities and poor performance. Protection mechanisms against such energy attacks are essential to realize the EAC vision.

Chapter 4

Coordinated Energy and Thermal Adaptation in Datacenters

This chapter explores cluster EAC in detail in the case of transactional workloads that are typically served by virtual machines. In particular, this chapter elaborately discusses the design of a control system called *Willow* [9, 10] to provide energy and thermal adaptation within a data center. *Willow* provides a hierarchical energy adaptation within data centers in response to both demand and supply side variations. Although a comprehensive energy adaptation scheme could include many facets, the current design of *Willow* is geared towards load consolidation and migration. The design and implementation of the control algorithms is explained in Sections 4.1 to 4.3 In Section 4.4 the design of a QoS aware scheduling scheme is presented. The proposed scheduling scheme complements *Willow* to achieve the guaranteed QoS for different applications in the presence of adaptation control actions.

4.1 Hierarchical Power Control

Power/energy management is often required at multiple levels including individual devices (CPU cores, memory DIMMs, NICs, etc.), subsystems (e.g., CPU - cache subsystem), systems (e.g., entire servers), and groups of systems (e.g., chassis or racks). In a power limited situation, each level will be expected to have its own power budget, which gets divided up into power budgets for the components at the next level. This brings in extra complexity since one must consider both the demand and supply sides in a coordinated fashion at various levels. This section describes such a multilevel power control architecture. One simple such power management model is shown in Figure 4.1. The data center level power management unit (PMU) is at the level 3. The rack level PMU is at level 2 and server/switch level PMUs are at level 1. With such a multilevel power management architecture the proposed control scheme attempts to provide the scalability required for handling energy and thermal adaptation in large data centers with minimum impact on the underlying networks.

In the hierarchical power control model that is assumed in this chapter, the power budget in every level gets distributed to its children nodes in proportion to their demands. All the leaf nodes are in level 0. The component in each level $l + 1$ has configuration information about the children nodes in level l. For example the rack level power manager has to have knowledge of the power and thermal characteristics of the

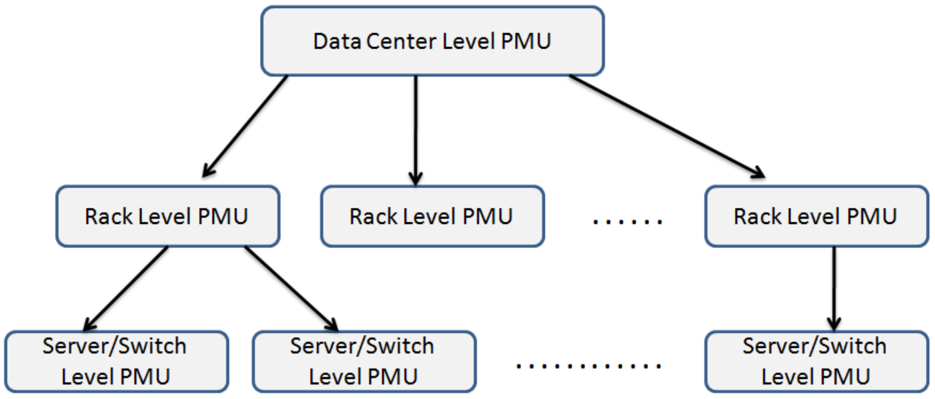


Figure 4.1: A simple example of multi-level power control in a datacenter

individual components in the rack. The components at level l continuously monitor the demands and utilization levels and report them to level $l+1$. This helps level $l+1$ to continuously adjust the power budgets. Level $l+1$ then directs the components in level l as to what control action needs to be taken. The granularities at which the monitoring of power usage and the allocation adjustments are done are different and are discussed later in Section 4.3.1.

4.2 Energy-Temperature relationship

In the design of *Willow* the power consumption of a device is limited based on its thermal limits as follows.

Let t denote time, $T(t)$ the temperature of the component as a function of time, $P(t)$ power consumption as a function of time, and c_1, c_2 be the appropriate thermal constants. Also, let T_a denote the ambient temperature, i.e., temperature of the medium right outside the component. The component will eventually achieve this temperature if no power is supplied to it. Then the rate of change of temperature is given by

$$dT(t) = [c_1 P(t) + c_2(T(t) - T_a)]dt \tag{4.1}$$

Being a first-order linear differential equation, this equation has an explicit solution. Let $T(0)$ denote the temperature at time $t = 0$. Then,

$$T(t) = [T_a + [T(0) - T_a]e^{-c_2 t}] + c_1 e^{-c_2 t} \int_0^t P(\tau) e^{c_2 \tau} \, d\tau \tag{4.2}$$

where the first term relates to cooling and tends to the ambient temperature T_a and the second term relates to heating. Let T_{limit} denote the limit on the temperature and P_{limit} is the limit on power consumption so that the temperature does not exceed T_{limit} during the next adjustment window of Δ_s seconds. It is easy to see that,

$$T(\tau) = T_a + P_{limit} c_1 / c_2 [1 - e^{-c_2 \Delta_s}] + [T(0) - T_a] e^{-c_2 \Delta_s} \tag{4.3}$$

It can be observed that Equation 4.2 can be used to predict the value of temperature of the device at the end of the next adjustment window and hence can help in making the migration decisions. This relationship is used to estimate the maximum power consumption that can be allowed on a node so that it does not exceed its thermal limits.

4.3 Supply And Demand Side Coordination

In *Willow* migrations of power demands are initiated by the power and thermal constraints introduced as a result of increase in demand at a particular node or decrease in power budget to the node. Simultaneous supply and demand side adaptations are done to match the demands and power budgets of the components.

4.3.1 Time Granularity

The utilization of server resources in a data center varies widely over a large scale. If the nature of the workload fluctuates significantly, it is likely that different resources (e.g., CPU cores, DRAM, memory bus, platform links, CPU core interconnects, I/O adapters, etc.) become bottlenecks at different times; however, for a workload with stable characteristics (but possibly varying intensity) and a well-apportioned server, there is one resource (typically CPU and sometimes network adapter) that becomes the first bottleneck and its utilization can be referred to as server utilization. This assumption forms the basis for the modeling presented in this chapter, since it is extremely difficult to deal with arbitrarily configured servers running workloads that vary not only in intensity but their nature as well. Under these assumptions, the power consumption can be assumed to be a linear monotonic function of the utilization.

Because of varying intensity of the workload, it is important to deal with average utilizations of the server at a suitable time granularity. For convenience the demand side adaptations are discretized with a time granularity of Δ_{Dl}. It is assumed that this time granularity is sufficiently coarse to accommodate accurate power measurement and its presentation, which can be quite slow. Typically, appropriate time granularity at the level of individual servers are of the order of tens of milliseconds or more. Coarser granularities may be required at higher levels (such as rack level).

Even with a suitable choice of Δ_{Dl}, it may be necessary to do further smoothing in order to determine trend in power consumption. Let $CP_{l,i}$ be the power demand of node i at level l. For exponential smoothing with parameter $0 < \alpha < 1$, the smoothed power demand CP' is given by:

$$CP'_{l,i} = \alpha CP_{l,i} + (1-\alpha) CP'^{old}_{l,i} \tag{4.4}$$

Note that the considerations in setting up the value of Δ_{Dl} come from the demand side. In contrast, the supply side time constants are typically much larger. Because of the presence of battery backed UPS and other energy storage devices, any temporary deficit in power supply in a data center is integrated out. Hence the supply side time constants are assumed to be $\Delta_{Sl} = \eta_1 \Delta_{Dl}$, where η_1 is an integer and $\eta_1 > 1$. *Willow* also performs workload consolidation when the demand in a server is very low so that some servers can be put in a deep sleep state such as S3 (suspend to memory) or even S4 (suspend to disk) [83]. Since the activation/deactivation latency for these sleep modes can be quite high, another time constant Δ_{Al} is used for making consolidation related decisions. It is assumed that $\Delta_{Al} = \eta_2 \Delta_{Dl}$, for some integer η_2 such that $\eta_2 > \eta_1$.

4.3.2 Supply Side Adaptation

As mentioned earlier the case where the data center operates in a perpetually energy deficient regime, is ignored. The available power budget of any level $l+1$ is allocated among the nodes in level l proportional to their demands. As mentioned in Section 4.3.1

the supply side adaptations are done at a time granularity of Δ_{Sl}. Hence the power budget changes are reflected at the end of every Δ_{Sl} time period. Let TP_{l+1}^{old} be the overall power budget at level $l+1$ during the last period. TP_{l+1} is the overall power budget at the end of current period. $\Delta_{TP}= \mathrm{TP}_{l+1} - \mathrm{TP}_{l+1}^{old}$ is the change in overall power budget. If Δ_{TP} is small, the values of $\mathrm{TP}_{l,i}$'s can be updated rather trivially. However if Δ_{TP} is large, the power budgets of nodes in level l need to be reallocated. In doing so *Willow* considers both *hard* constraints due to power limitations of devices and *soft* constraints due to available power budgets.

The power and thermal constraints thus necessitate the migration of demand in level l from power deficient nodes to nodes with surplus power budget. Any increase in the overall power budget happens at a higher level and is then reflected in its constituent lower levels. This situation can lead to three subsequent actions.

1. If there are any under provisioned nodes they are allocated just enough power budget to satisfy their demand.
2. The available surplus can be harnessed by bringing in additional workload.
3. If surplus is still available at a node then the surplus budget is allocated to its children nodes proportional to their demand.

4.3.3 Demand Side Adaptation

The demand side adaptation to thermal and energy profiles is done systematically via migrations of the demands. It is assumed that the fine grained power control in individual nodes is already being done so that any available idle power savings can be harvested. The main focus in this chapter is on workload migration strategies to adapt to the energy deficient situations. For specificity only those type of applications in which the demand is driven by user queries and there is minimum or no interaction between servers, (e.g., transactional workloads) are considered. The applications are hosted by one or more virtual machines (VMs) and the demand is migrated between nodes by migrating these virtual machines. Hence the power consumption is controlled by simply directing the user queries to the appropriate servers hosting them.

Willow carefully avoids pitfalls like oscillations in decisions by allowing sufficient margins both at the source and the destination to accommodate fluctuations after the migrations are done. The migrations are initiated in a bottom up manner. If the power budget $\mathrm{TP}_{l,i}$ of any component i is too small then some of the workload is migrated to one of its sibling nodes. This is termed as local migration. Only when local migrations to sibling nodes is not possible, non–local migrations are done.

The migration decisions are made in a distributed manner at each level in the hierarchy starting from the lowermost level. The local demands are first satisfied with the local surpluses and then those demands that are not satisfied locally are passed up the hierarchy to be satisfied non-locally. A few terms related to the migration decisions are defined below.

Power Deficit and Surplus: The power deficit and surplus of a component i at level l are defined as follows.

$$P_{def}(l,i) = [CP'_{l,i} - TP_{l,i}]^{+} \tag{4.5}$$

$$P_{sur}(l,i) = [TP_{l,i} - CP'_{l,i}]^{+} \tag{4.6}$$

where $[]^{+}$ means if the difference is negative it is considered zero.

From the above definitions the power surplus and deficit at a particular level can be defined as follows.

$$P_{def}(l) = max(P_{def}(l,i)) \tag{4.7}$$

$$P_{sur}(l) = max(P_{sur}(l,i)) \tag{4.8}$$

If there is no surplus that can satisfy the deficit in a node, the excess demand is simply dropped. In practice this means that some of the applications that are hosted in the node are either shut down completely or run in a degraded operational mode (e.g., higher latency, lower resolution etc.) to stay within the power budget.

Power Margin (P_{min}): The minimum amount of surplus that has to be present after a migration in both the source and target nodes of the migration. This helps in mitigating the effects of fluctuations in the demands.

Migration Cost: The migration cost is a measure of the amount of work done in the source and target nodes of the migrations as well as in the switches involved in the migrations. This cost is added as a temporary power demand to the nodes involved.

A migration is done if and only if the source and target nodes can have a surplus of at least P_{min}. Also migrations are done at the application (VM) level and hence the demand is not split between multiple nodes. Finally *Willow* also does resource consolidation to save power whenever possible. When the utilization in a node is really small the demand from that node is migrated away from it and the node is deactivated.

Even with the above mentioned constraints there are multiple ways of matching a demand with a surplus. The entire problem now reduces to the classical bin packing problem. The surpluses available in different nodes form the bins. The bins are variable sized and the demands need to be fitted in them. The variable sized bin packing problem is a NP-hard problem as such and numerous approximation schemes are available in literature [84, 85, 86]. One such simple scheme called FFDLR [86] was chosen to solve the bin packing problem. Let L be the list of demands that are to be satisfied. The FFDLR algorithm is described in Algorithm 1.

Algorithm 1 FFDLR Algorithm

```
  Step 1: The bin sizes and demand sizes are normalized so that the largest bin has a
  size of 1.
  Step 2: Pack the first demand in $L$ into the first bin of size of 1 that has room for it
  and remove the demand from list $L$.
  Step 3: Repeat Step 2 until all demands are packed in a bin.
  Step 4: At the end, the contents of each bin are repacked into the smallest possible
  empty bins.
```

FFDLR solves a bin packing problem of size n in time $O(n \, log \, n)$. The optimality bound guaranteed for the solution is $(3/2) \, OPT + 1$ where OPT is the solution given by an optimal bin packing strategy. This algorithm was chosen for two reasons. First, it is simple to implement with guaranteed bounds. Second, the repacking into smaller bins means every server will run close to full utilization. The servers that are empty can then be deactivated during consolidation phase.

4.4 QoS Aware Scheduler

Traditional scheduling algorithms like round robin and priority based scheduling are not QoS aware and do not have any feedback mechanism to guarantee the QoS requirements of jobs. A few algorithms that attempt to guarantee some level of QoS do so by mechanisms like priority treatment and admission control. The objective in this thesis work is to allow for energy adaptation while respecting QoS needs of various applications to

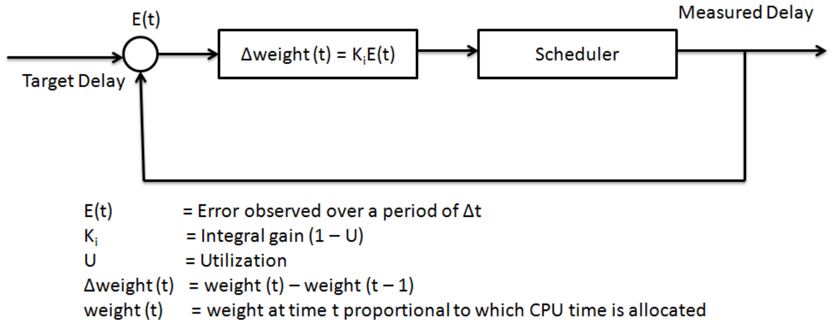

E(t)
Measured Delay
Δweight (t) = K_iE(t)
Scheduler
Target Delay
E(t) = Error observed over a period of Δt
K_i = Integral gain (1 – U)
U = Utilization
Δweight (t) = weight (t) – weight (t – 1)
weight (t) = weight at time t proportional to which CPU time is allocated

Figure 4.2: QoS Aware Scheduler

the maximum extent possible. In this regard, a QoS aware scheduling algorithm in the nodes is proposed in this section and is shown in Figure 4.2. The scheduler uses an integral controller to adjust the weights of the applications at regular intervals based on the delay violations of the applications. The applications are allocated CPU shares proportional to their weights. Applications with higher delay violations get more CPU share. It is well known from classic queuing theory [87] that in an asymptotic sense, as $U \rightarrow 1$ the wait time of jobs is directly proportional to $1/(1-U)$ where U is the overall queue utilization. Hence the integral gain for the controller is calculated as $(1-U)$. Using the overall utilization as the integral gain also avoids oscillations and keeps the system stable.

The error in delay $E(t)$ for each application is the difference between the delay bound and the measured delay. At the end of each sample interval t (of size Δt), the new weights are calculated as follows.

$$weight(t) = weight(t-1) + K_i * E(t) \tag{4.9}$$

Then the CPU shares are allocated to the applications proportional to their weights.

Figure 4.3 shows an overall picture of the different adaptation mechanisms that are done in *Willow* at different time granularities. The scheduler works in the individual nodes at the smallest time granularity. At the next higher time granularity (Δ_{Dl}), the demand side adaptations are done that include migration of deficits to surplus nodes. At the next higher granularity (Δ_{Sl}), the supply side adaptations such as allocating power budgets at different levels takes place. At the largest time granularity (Δ_{Al}), consolidation related decisions are made that include shutting down of nodes with very low utilization.

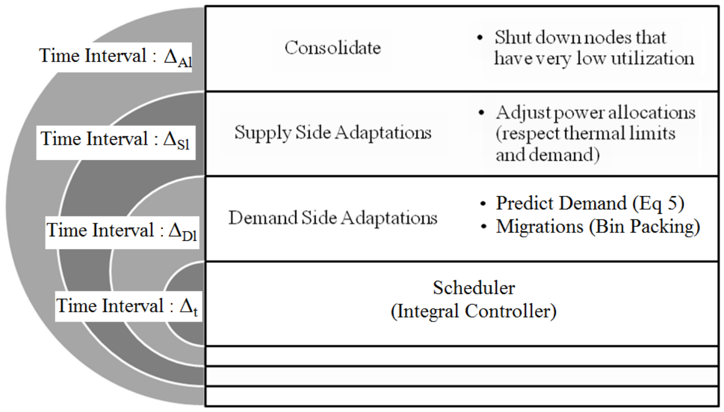

Time Interval : Δ_{Al}
Time Interval : Δ_{Sl}
Time Interval : Δ_{Dl}
Time Interval : Δ_t
Consolidate
• Shut down nodes that have very low utilization
Supply Side Adaptations
• Adjust power allocations (respect thermal limits and demand)
Demand Side Adaptations
• Predict Demand (Eq 5)
• Migrations (Bin Packing)
Scheduler
(Integral Controller)

Figure 4.3: Various adaptations done in *Willow* and the different time granularities

4.5 Experimental Evaluation

This section presents some experimental results to demonstrate the ability of the proposed control scheme (*Willow*) to cater to the QoS requirements of tasks when there are energy variations. *Willow* was evaluated via detailed simulations using a Java simulator and real time experiments on a prototype testbed. The simulations were based on generalized assumptions on the power demands and limits. The experiments were done to test the working of *Willow* when real migrations of VMs is done practically. The simulations and experiments demonstrate the adaptability of *Willow* to varying energy and thermal conditions. Section 4.5.1 explains the preliminary experiments on a prototype cluster and Section 4.5.2 explains the simulation results that analyze the performance and behavior of *Willow* in a more detailed manner.

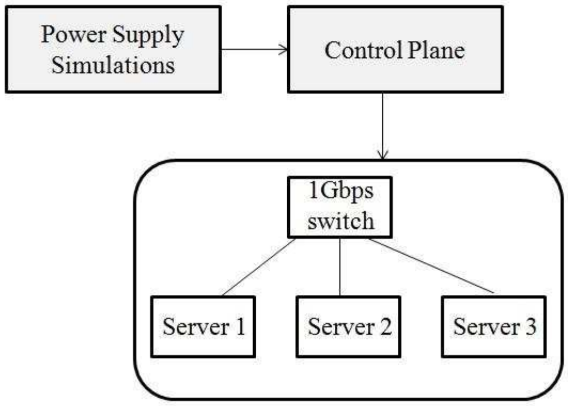


Figure 4.4: Experimental Testbed

Table 4.1: Utilization vs power consumption

Utilization%	Average Power consumed(Watts)
20	175
40	190
60	215
80	230
100	245

4.5.1 Preliminary Experiments

Experimental Testbed

Experiments to evaluate the performance of *Willow* were done on a prototype testbed consisting of 3 servers. Figure 4.4 shows the architecture of the experimental testbed. A cluster of three Dell servers was used. VMWare ESX server 3.5 [88] was running on all three of the servers. The three servers were managed from a remote machine. The remote machine forms the control plane which simulates the hierarchical power control model and the power supply variations. Since there were only 3 machines, the power–control hierarchy was simulated with two levels – two switches in level 1 and one switch and the three servers in level 2. CPU temperature of the servers was measured from

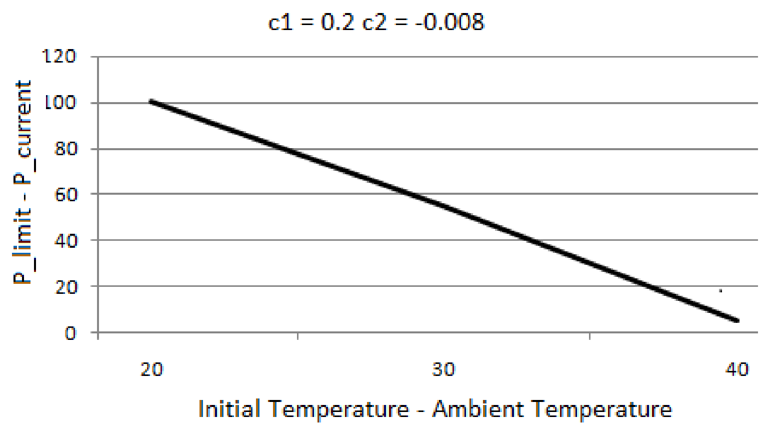


Figure 4.5: Experimental estimation of parameters c_1 and c_2

the onboard sensor. CPU utilization of the ESX servers was monitored continuously by a script running on the control plane. The variation in power supply was introduced into the system artificially and the control system made migration decisions to adapt to the variations in power supply. Virtual machines were installed on each of the ESX servers. These virtual machines were hosting web servers that supported one of three applications, each with different power profiles. For simplicity all the applications were CPU bound. Hence the power–temperature measurements that are described in this section correspond to CPU utilization.

Baseline Experiments

Baseline experiments were conducted to parameterize the Equation 4.1 and to capture the relationship between power and utilization. A CPU intensive application was running in the server. Table 4.1 shows the power consumption of one of the servers for various CPU utilizations. It can be seen that the power consumption is a continuously increasing function of utilization. The static power consumption was found to be almost a constant. With only a single component in the server (in this case CPU) being involved in the computations, this kind of a linear relationship is quiet intuitive. The scenario where power consumption follows only the utilization is an idealistic situation because that means the idle power consumption is zero. Fine grained power management techniques aim at achieving this goal by trying to minimize the static power consumption. The scope of *Willow* is not idle power control but to adapt to the power supply and demand. The power consumption was measured using an Extech 380801 power analyzer [89]. The sampling rate of the power analyzer was 2 Hz.

Figure 4.5 shows the trend in power consumption vs temperature. The parameter values in Equation 4.1 were determined by measuring the temperature with the power consumed and the temperature constants were determined to be $c1 = 0.2$ $c2 = -0.008$. The y–axis in Figure 4.5 is the maximum power that can be accommodated (available power surplus) and the x–axis is the difference between ambient temperature and current temperature of the device. The ambient temperature was assumed to be uniform throughout the room and was equal to 25° C.

Application Profiling

Table 4.2: Application Power Profile

Application	Increase in power consumption(Watts)
A_1	8
A_2	10
A_3	20

Three different applications A_1, A_2 and A_3 with different CPU utilizations and hence different power consumptions were used for the experiments. Table 4.2 shows the increase in power consumption when three applications are running on the server. Each application is run on a single virtual machine. The power consumption of the servers can be controlled by migrating the VMs away from or to the servers.

Experimental Results

This section explains the observed results when the servers are running at an overall average utilization level of 60%. Figure 4.6 shows the power supply variation pattern that was used in the simulation when the average utilization is 60%. As stated before,

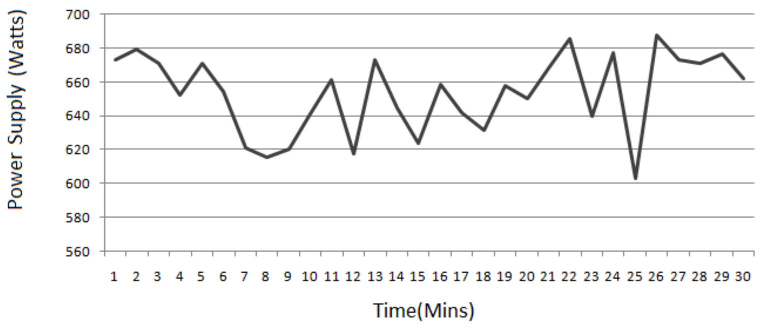


Figure 4.6: Power supply variation (Energy Deficient Situation)

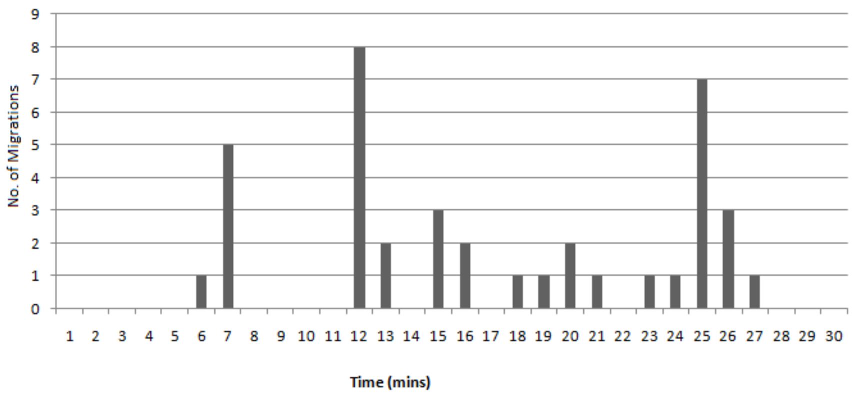


Figure 4.7: Number Of Migrations

the perpetually energy deficient regime is simply ignored and only short term energy deficiencies are considered. The available power supply is divided proportionally between the servers. When the available power supply is reduced the tasks are migrated away from servers running at high utilizations to servers running at a low utilization. In general the number of migrations to or from a server depends on the available power supply and the utilization at which it is running. This can be observed in Figure 4.7. For instance, in Figure 4.6 at time unit 7 the power consumption plunges deeply. In a power deficient situation, servers that are running at a higher utilization have to migrate some workload away from them to maintain the same QoS. For simplicity it is assumed that the QoS of applications is a function of the power demand of the node and as the power demand (load) increases, the application perceived QoS is assumed to degrade.

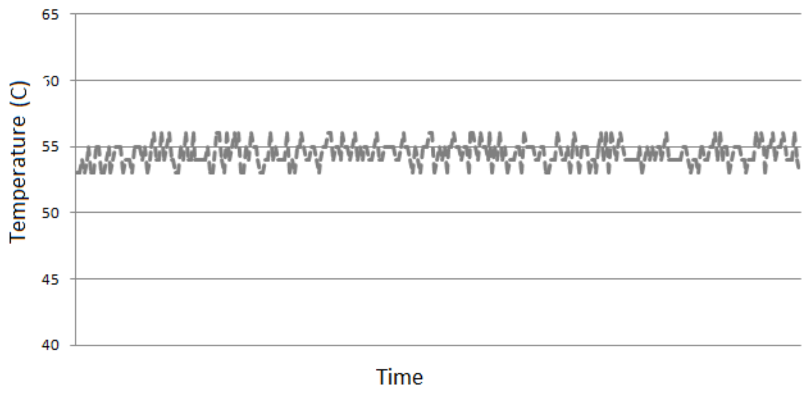


Figure 4.8: Temperature Time Series (Server A)

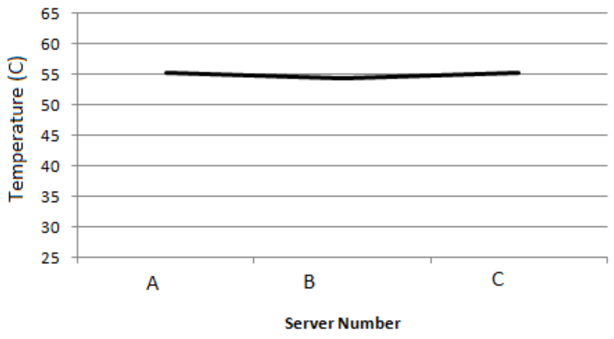


Figure 4.9: Average Temperature of Servers

This relationship is modeled in a more sophisticated manner later in Section 4.5.2.

It can be seen from Figure 4.7 that the number of migrations is also high in time unit 7. The decreased power supply persists till time unit 10. However there are no migrations between time units 7 and 10. This is because, once the migrations are done there is enough margin left to handle the demand variations. Hence no more migrations are needed. Similar patterns can be observed in time units 12 and 25. Note that after a deep plunge in power supply there is not much migration activity even if the power supply increases(e.g. time units 25–26). This is because of the fact that the migrations in *Willow* are mainly initiated by the tightening of power constraints and not by their loosening. The initiation of migrations in a power–plenty situation can happen only at very low utilization levels where there is a possibility of shutting down a server completely.

Power Savings

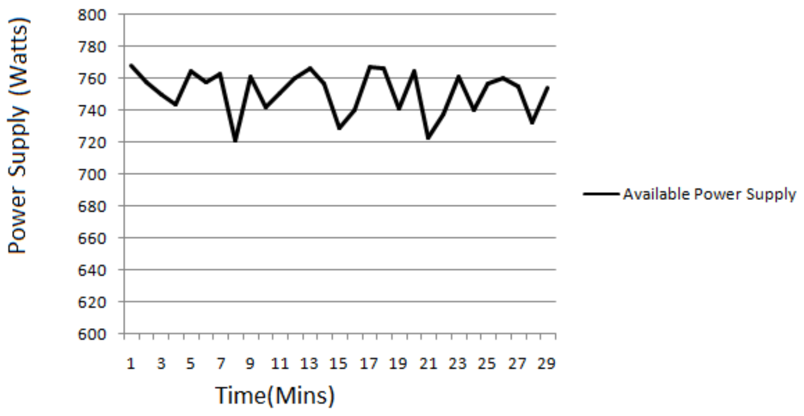


Figure 4.10: Power supply variation(Energy Plenty situation)

This section describes the potential power savings achieved by *Willow* when there is a large surplus in power supply. In other words this scenario is the case where some servers are running at low utilization levels. The available power supply varied as in Figure 4.10. Note that the average power supply available is close to the necessary power supply to support the three servers at a utilization level of 100%($\approx 750W$). Consolidation driven migrations are initiated when the utilization in the server is lower than a migration threshold. The migration threshold was set to be 20%. Table 4.3 shows the initial utilization levels of the servers and average utilization levels of servers after the run of the experiments.

It can be seen that the utilization of Server C is 0%. This is because Servers A and B are running below their power and thermal limits even after the tasks have been migrated from Server C. Hence there is no need to wake up server C. Server C remains shut down for the entire run of the experiments.

Without consolidation, if the servers A, B and C were running at utilizations of 80%,40% and 20% respectively, their average power consumption from Table 4.1 is $(175 + 190 + 215 =)$ 580 W. However after consolidation, the average power consumption is $(230 + 190 + \text{standby})$ $420 + \text{standby}$–power consumption for server C. It is reasonable

Table 4.3: Utilization of Servers

Server	Initial Utilization%	Average Utilization at the end of experiment
A	60	80
B	40	40
C	20	0

to assume that this stanby power consumption is very negligible. For instance, VMWare ESX server has the Dynamic Power Management [90] utility that can shut down an idle host and the power consumed is zero. With that assumption the power savings achieved is around 27.5%.

4.5.2 Simulation

Assumptions and QoS model

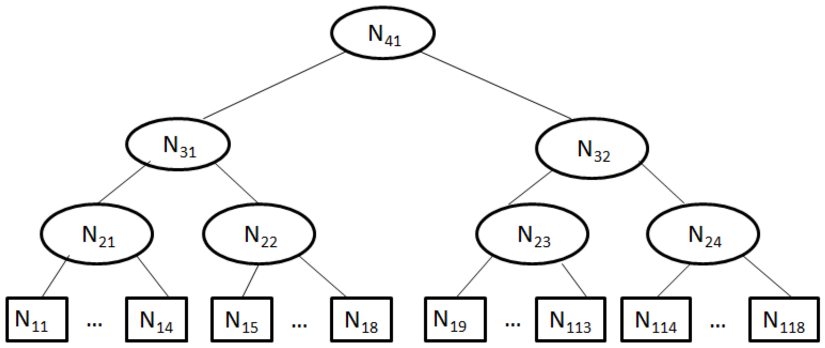


Figure 4.11: Configuration used in simulation

Table 4.4: Utilization vs power consumption

Application Type	SLA Requirement	Mean Runtime
Type I	Average Delay $\leq 120ms$, cannot be migrated	$10\,ms$
Type II	Average Delay $\leq 180ms$, can be migrated	$15\,ms$
Type III	Average Delay $\leq 200ms$, can be migrated	$20\,ms$

Since the real time experiments were constrained by the number of servers, simulations were done to analyze the performance of *Willow* in a more detailed manner. The Java simulator can be configured to simulate any number of nodes and levels in the hierarchy. The configuration used in the evaluations is shown in Figure 4.11. There are 18 nodes and 3 types of applications. The application types and their SLA requirements are shown in Table 4.4. For simplicity, it is assumed that each application is hosted in a VM and can be run on any of the 18 nodes. This also helps in migration of workloads between nodes since virtual machines can be easily migrated from one server to another. To begin with, each node is assigned a random mix of applications. The static power consumption of nodes is assumed to be 20% of the maximum power limit ($450W$). There is a fixed power cost associated with migrations. In the configuration that was used in the experiments it is assumed that there is a single SAN storage that can be accessed by all nodes in the data center. Storage migration is done when the VM disk files have to be migrated across shared storage arrays due to shortage of storage capacity and is usually dealt with separately (eg. Storage VMotion [91]) so as to reduce the delays involved. This section deals only with compute intensive applications and it is assumed that a single shared storage domain is large enough to support the applications and the delays involving data migrations are not accounted for.

Simulation setup and Results

The experimental platform consists of multiple VMs with independent traffic patterns. Soccer World Cup 98 [92] traces were used for evaluation. The Soccer World Cup 98 trace dataset consists of all the requests made to the 1998 Soccer World Cup Web site during April to July, 1998. Since the trace data was collected from multiple physical servers, the traces were pre–processed in order to be used in the simulated, scaled down virtualized environment.The traces collected on different days were used on different VMs. The arrival rates of queries were scaled appropriately to achieve the target utilization levels. It is assumed that the queries in the traces belonged to the three classes of applications as shown in Table 4.4. The service times for queries for each application class is different and the energy consumed by a query is assumed to be proportional to its runtime. The time constant multipliers for discrete time control η_1 and η_2 in Section 4.3.1 are assumed to be 4 and 7 respectively.

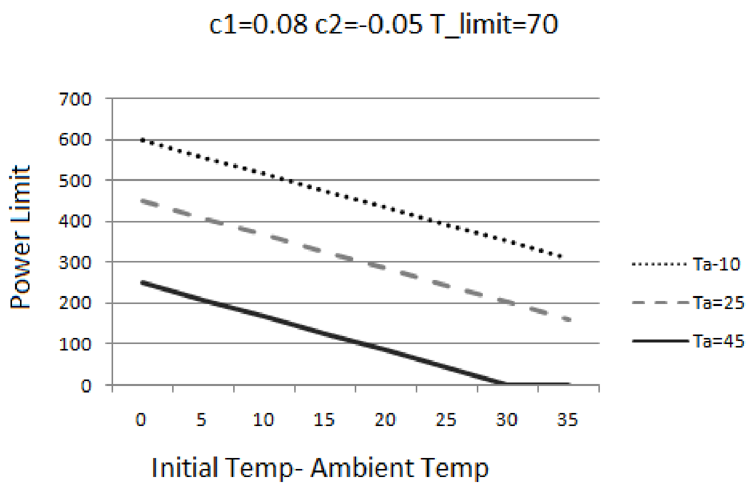


Figure 4.12: Setting up the thermal constants

The thermal constants in Equation 4.1 are dependent on the thermal properties of the individual devices. For the simulation the values of these constants are calculated by assuming reasonable values for the maximum device power and the ambient temperature. It is assumed that the average server/switch power consumption is around 450 Watts and a typical ambient temperature to be 25°C. Also the thermal limit of the servers and switches is assumed to be 70°C. With these settings, Figure 4.12 shows one possible set of values for the constants c_1 and c_2. When the power consumption is zero (when the server is in a deep sleep state because of no activity), the component is at the ambient temperature. The power surplus that is presented at this temperature should be approximately the same as the maximum power rating of the component. From Figure 4.12 it can be seen that the values $c_1 = 0.08$ and $c_2 = -0.05$ show the maximum power limit to be around 450 W when the ambient temperature T_a is 25°C and the initial power consumption is zero. Hence these values were chosen for c_1 and c_2 for the experiments. It can be seen from Figure 4.12 that when the ambient temperature T_a=45°C and the temperature of the server is at 70°C the power surplus that is presented is almost zero because the server has already reached its thermal limit. It is assumed that the time taken for the increase in temperature when the demand increases, is less than Δ_{Dl}. This is a conservative assumption. In reality the increase in temperature is not instantaneous. However such an assumption helps to reduce the number of migrations and instability in decisions.

The utilization of a VM is measured based on CPU time spent by the VM in servicing the requests. However the actual utilization may be higher. For instance the actual utilization of a task may be increased due to CPU stalls that are caused by memory contention from other tasks or context switches between multiple VMs. In the simulations, a factor called the interference penalty for the utilization and power/energy calculations is included. Basically the idea is that when n tasks are running on a server, the utilization of each task is increased due to the interference from the other $(n-1)$ tasks. Hence the actual utilization of an application is calculated as follows.

$U_i = U_i + \alpha \sum_j U_j$, $\forall j \in \{1, 2, ...n\} - \{i\}$

$U_{node} = \min\,[1.0,\, \sum_{i=1}^{n} U_i]$

where n is the total number of applications in the node

U_i is the utilization of i^{th} application , $i \in \{1, 2, ...n\}$.

U_{node} is the actual utilization of the node.

the average power consumed in the node is then calculated based on the actual average utilization of the node. A few experiments were conducted on a Dell machine running VMWare ESX server to determine the value of α. The number of VMs and their utilization levels were varied and the sum of their utilizations was compared with the actual utilization reported by the ESX server. The value of α was then calculated using simple first order linear regression. The value of α was found to be 0.01. It is to be noted that the workload that was used in the experiment was totally CPU bound. For other workloads that involve memory or network contention, the interference penalty might be different (perhaps higher).

In order to evaluate the performance of the integral controller all 3 types of applications were placed on a single node and the incoming traffic to each of the applications was assumed to be a Poisson process. Figure 4.13 shows the delays of queries normalized to their delay bounds at a utilization level of 60%. Any value of delay greater than 1 implies an SLA violation. A QoS ignorant scheduler simply serves queries as they arrive. This leads to increased delay violations for queries. For instance, when there are too many Type I queries that have a smaller delay bound waiting in the queue, a QoS aware scheduler will make Type III queries to wait a little longer in the queue since they can tolerate much longer delays. On the other hand a QoS ignorant scheduler continues to favor both Type I and Type III queries equally and hence leading to delay violations for Type I queries. It can be seen from Figure 4.13 that almost during the entire time, the feedback based integral controller successfully keeps the delays of all 3 types of queries below the bounds specified by their SLA requirements.

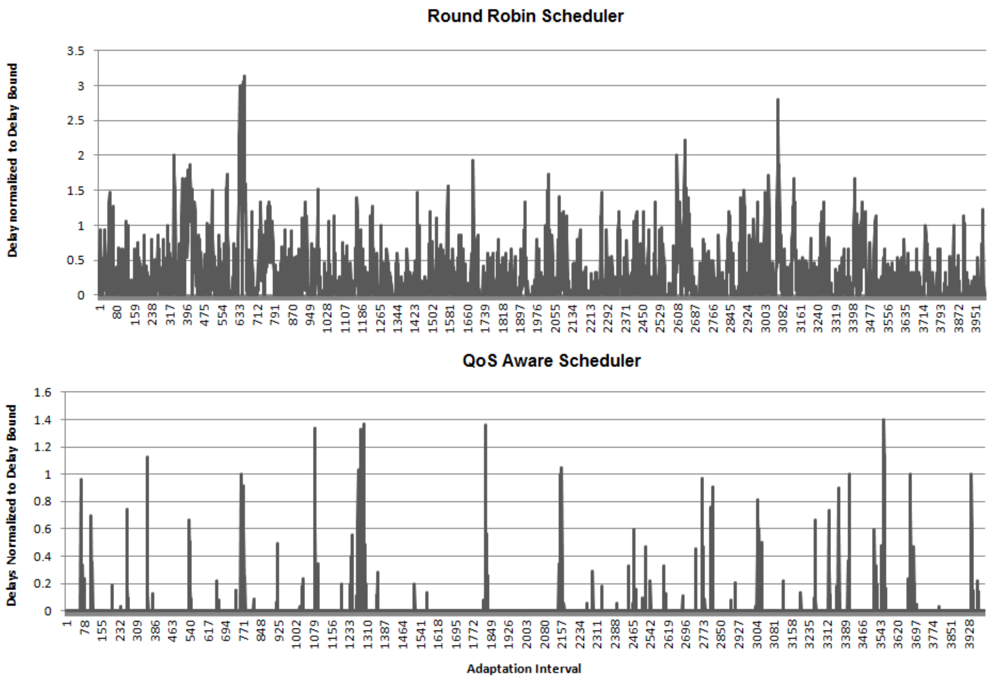


Figure 4.13: Time series of average delays of queries normalized to delay bounds with simple Round Robin and QoS Aware Scheduling

The response time is used as a metric to quantify the impact of EAC in the presence of variations in energy. The response time includes the time that the queries wait in the queue to be scheduled and the run time of the queries. The significance of Willow is realized the most when the devices are operating at the edge – that is when the power budgets are just enough to meet the aggregate demand in the data center and there is no over provisioning. To demonstrate this, the performance of Willow was tested with two different cases of simulated power budgets. Let P_U be the power required to support the data center operations when the average utilization of the servers is $U\%$. Figure 4.14 shows the histogram of the total available power budget values during the simulation of 350 minutes and the proportion of time for which the particular power budget value was available. The first case (Case 1) is when the total available power budget varies between P_{100} and P_{60}. The second case (Case 2) is when the total power budget varies between t P_{80} and P_{50}.

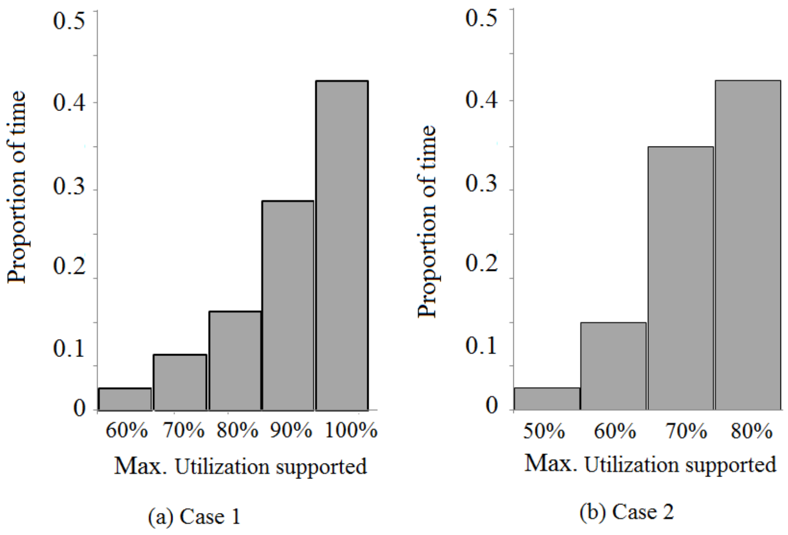
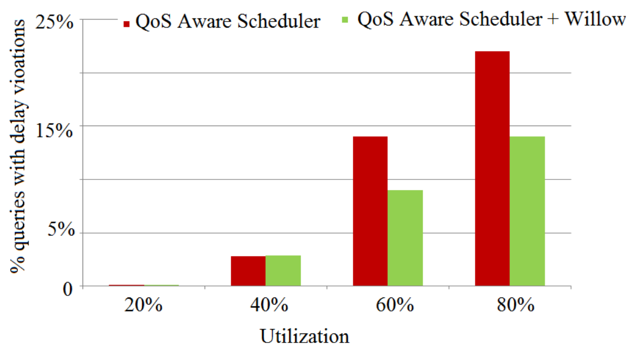
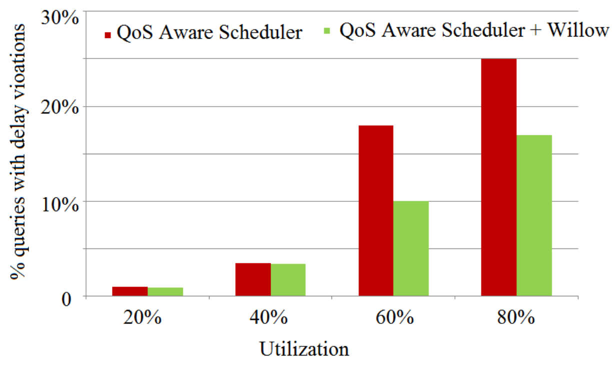


Figure 4.14: Power Supply Profiles Used - Histogram of available power supply and the maximum utilization levels supported

Figure 4.15 (a) compares the percentage of queries with delay violations in Case 1 when the QoS aware scheduler alone is used and when the scheduler is used in combination with *Willow*. It can be seen that at low utilizations the performance of Willow is not significantly better than when the QoS aware scheduler alone is used. However at high utilizations *Willow* performs better than the case when the QoS aware scheduler alone is used since adaptation related migrations done in *Willow* help to reduce the effect on the QoS of applications. Figure 4.15 (b) shows the percentage of queries with delay violations in Case 2. It can be seen that the benefits of Willow are very significant in Case 2 as compared to Case 1, especially at moderate to high utilization levels. Figure 4.15 shows that an efficient QoS aware scheduler alone cannot do any good in the presence of energy variations. *Willow* improves the possibility of meeting the QoS requirements significantly with the help of systematic migrations.



(a)



(b)

Figure 4.15: % of queries with delay violations when the power profile used is as shown in (a) Case 1 of Fig. 4.14 (b) Case 2 of Fig. 4.14

Figure 4.16 shows the average delay of different application types normalized to their delay bounds at different utilization levels for power profiles as in Case 2. As expected the average delays increase with increase in utilization due to increased wait times. It can be seen that at higher utilizations, the average normalized delay is higher for Type I queries. The reason is twofold. Firstly, according to the SLA requirements of Type I queries in Table I, they cannot be migrated. This reduces the flexibility for these applications when the available power budget in the server becomes very low. Secondly, the absolute values of the delay bound is much smaller than the other two types and hence the average delay when normalized to the delay bound is naturally higher. Since the delay requirements of Type II and Type III are almost the same, they are equally

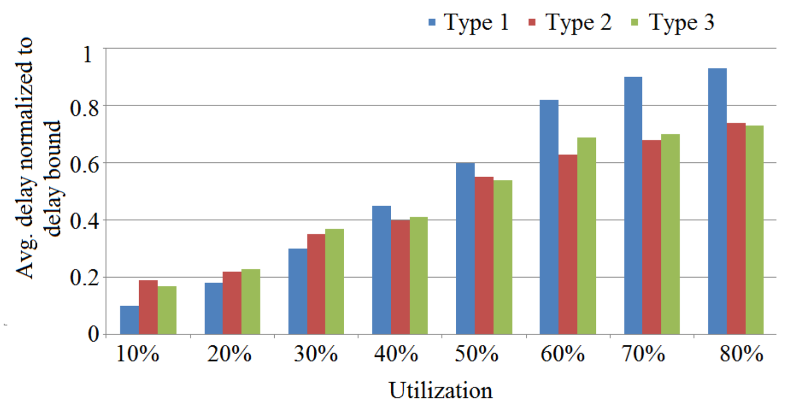


Figure 4.16: Average delay values normalized to delay bounds

impacted.

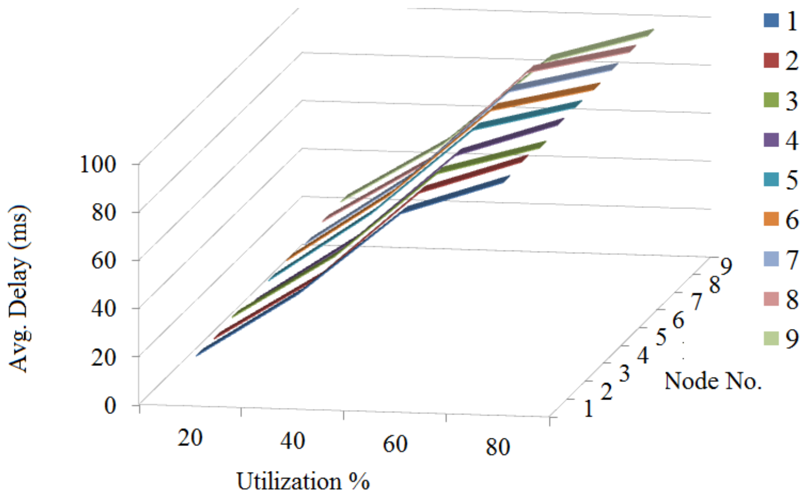
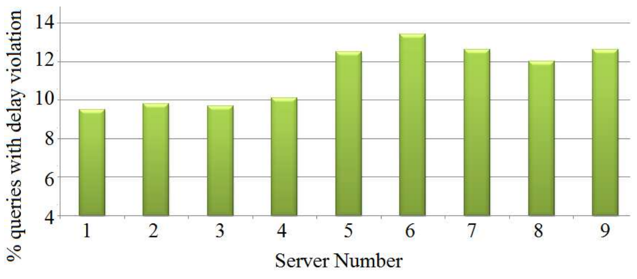
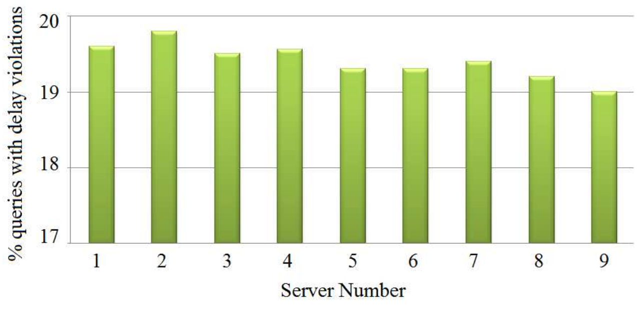


Figure 4.17: Avg. delay values for Type I queries when servers 1 − 4 are at a higher ambient temperature

It can be seen that even at low utilizations there are queries that have delay violations. This is because when the available energy drops, the demand side adaptations are done only after Δ_{Dl} in *Willow*. During that time if the estimated demand is less than the actual demand some queries have to wait longer to be scheduled. Moreover the utilization levels are just an average (calculated at a much higher granularity) and there can be periods of congestion even at those low utilization levels. This is especially



(a)



(b)

Figure 4.18: % of queries with delay violations when servers 1–4 are at high temperature at (a) 60% utilization (b) 80% utilization

handled very poorly when there are no migrations.

Figures 4.17 and 4.18 demonstrate the thermal adaptability of *Willow*. The ambient temperatures of servers $1 - 4$ was set at 45°C and the other servers at 25°C. In order to enforce power budgets, the available CPU shares were reduced proportional to their allocated power budgets. Figure 4.17 shows the average delays in the servers when the first four servers are at a higher temperature than others. To avoid clutter the average response times are shown only for the first 9 servers. As expected in all servers the response time increases with increase in utilization. However at low to moderately high utilization levels ($\leq 60\%$) it can be seen that the average response time is lower in high–temperature severs. This is because *Willow* is able to migrate most of the load away from high temperature servers. Hence the waiting time for queries is reduced which in turn improves the response times in the high–temperature servers.

Figure 4.18 (a) shows the percentage of queries with delay violations when the ambient temperature of servers 1–4 is 45°C at 60% utilization. As explained before, at a moderately high utilization level (60%), Willow migrates applications away from high temperature servers and hence they run at lower utilizations. This in turn reduces the number of queries with delay violations. However at high utilization (80%) the high temperature servers have no choice but to increase the delay of queries because no migrations can be done. This is shown in Figure 4.18 (b) .

Chapter 5

Energy adaptation in Multi-tiered Datacenters

This chapter investigates a specific incarnation of *EAC*, which is the manifestation of *Energy Adaptive Computing* in datacenters hosting multi–tiered web services. These include a wide variety of applications ranging from e–commerce services to large scale web search applications. Typically, the web service is hosted in multiple datacenters that are located in different geographical locations.

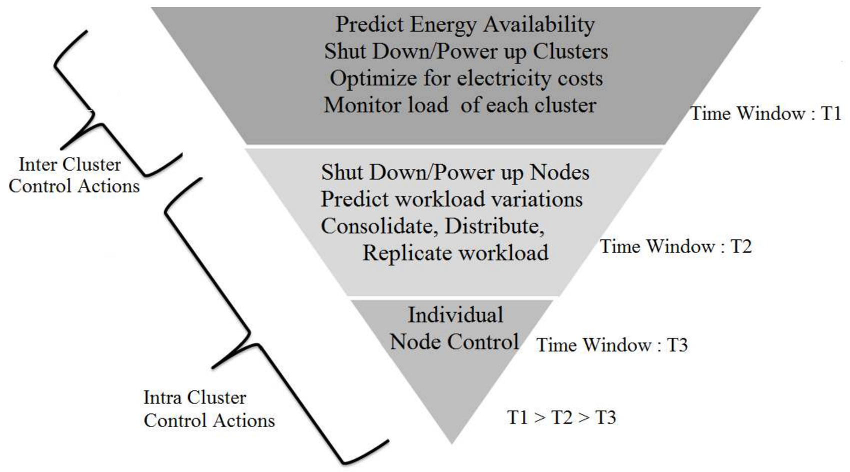

Predict Energy Availability
Shut Down/Power up Clusters
Optimize for electricity costs
Monitor load of each cluster
Time Window : T1
Inter Cluster Control Actions
Shut Down/Power up Nodes
Predict workload variations
Consolidate, Distribute, Replicate workload
Time Window : T2
Individual Node Control
Time Window : T3
Intra Cluster Control Actions
T1 > T2 > T3

Figure 5.1: Time windows for different power control actions

Figure 5.1 shows a scenario with different time constants in which the power control actions are done for an example datacenter with multiple clusters in different geographical locations. The time interval between successive control actions decreases as one moves down the pyramid. The control actions can be classified into two, based on the direction of initiation of the control actions–those that are initiated from the top–down and those from the bottom–up along the hierarchy. The largest time window $T1$ in Figure 5.1 represents the time interval during which the power supply varies. The variations are significant enough to initiate the control actions. The control actions may include predicting the available power for the next control period and migrating load from energy deficient clusters. In the multi–tier web server scenario that is being considered in this chapter, assuming that the services are stateless, workload migration involves redirecting more traffic to data centers with surplus energy. The number of servers in each cluster needs to be adjusted depending on the available power. Workload needs to be redistributed based on the number of servers that are kept powered on after the execution of the control actions. These are the top–down actions based on the available power. The bottom–up control actions are initiated from the node level. For instance some of the nodes might experience thermal constraints due to inefficient cooling or high thermal output. The demand variations and thermal constraints of the nodes need to be continuously monitored and reported to the managing entity (e.g., tier level load dispatcher) and the load and power budgets of the nodes need to be adjusted accordingly. This happens at a smaller time granularity ($T2 < T1$) than the power variations. At an even smaller time granularity ($T3 < T2$), the control actions may include adjusting the operational frequency of individual nodes or putting the nodes in shallow sleep modes if available.

This chapter proposes elaborate control actions that need to be coordinated in a single datacenter hosting a multi–tier web server application. The number of servers that are kept powered on and running in each tier is determined based on the available power and the delay constraints of the application. The expected delay with a given number of tiers in each tier is estimated based on queuing theoretic models and the best configuration that minimizes delay violations is chosen. Once this is done, some servers need to be powered on and some others need to be powered off. These operations are not instantaneous and involve significant overheads. For instance, before a server can be turned off, the pending queries that it was serving need to be completed and the workload that it was handling before needs to be redistributed to other servers. When a server that is currently powered off needs to be turned on, the activation period may be in the order of several seconds. An efficient planning of these operations is essential to avoid any adverse impacts on QoS. Many research works in the past have focused on formulating the power/performance trade–offs as optimization problems and have proposed solutions that would minimize these costs. This chapter presents an analytical design of an execution plan for the implementation of such optimization solutions. A load dispatcher at each tier adjusts the load allocated to different servers in the tier. All these actions are executed at different time granularities and help in adapting to the available power profiles and the workload variations for efficient operation of data centers.

5.1 Multi-tier Web Server Architecture

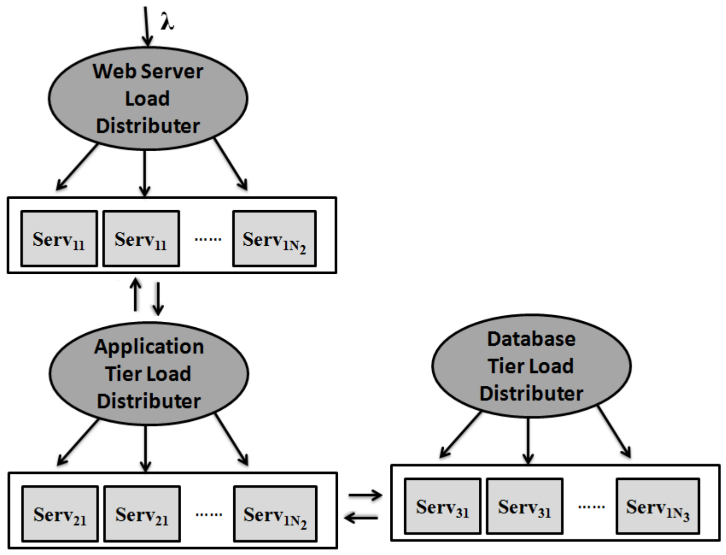

λ
Web Server Load Distributer
Serv11
Serv11
Serv1N2
Application Tier Load Distributer
Serv21
Serv21
Serv1N2
Database Tier Load Distributer
Serv31
Serv31
Serv1N3

Figure 5.2: Architecture of a three tiered datacenter

Figure 5.2 shows the typical architecture of a data center hosting a web service application like an e–commerce service with three tiers. The frontend nodes in Tier 1 process the HTTP requests and forward queries to the nodes in application tier (Tier 2). The application servers handle the queries and send the appropriate response back after communicating with the database tier (Tier 3) which consists of a clustered database handled by multiple nodes. The front end nodes then process the response and format the response (e.g., convert to HTML) to send it back to the user. Each query thus passes through the same tier multiple times. It is assumed that the servers in each tier are stateless from the application point of view and any server can process any query. However, when a query from the same session revisits a tier, it needs to be serviced by the same server that handled it before. The load in the datacenter depends on the number of sessions in progress and the number of servers involved in processing the queries. QoS guarantees are typically expressed in terms of the overall response time which is the time elapsed between the user request and the time when the user gets back the response. Recently, newer datacenter architectures have been proposed where a large number of cheap commodity servers (/switches) replace fewer powerful and expensive servers(/switches) [93, 94]. This is not only a power efficient architecture but also reduces the initial capital costs significantly. This chapter assumes that there are multiple commodity servers in each tier and are homogeneous. A load dispatcher in each tier assigns the queries to the appropriate servers based on their capacity and current load.

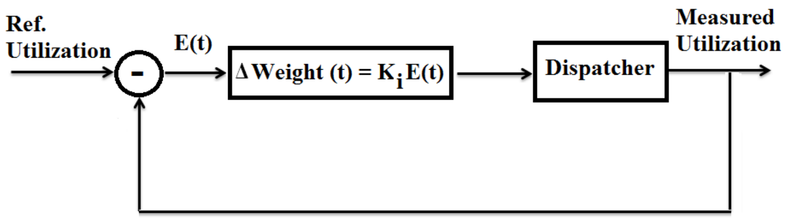


Figure 5.3: Integral Controller design for load dispatcher

5.1.1 Load Dispatcher

The service rate μ_j^i (of server j in tier i), is different for different servers depending on power budgets and thermal constraints. For instance even with sufficient power budgets certain servers can only be operated at half their capacity due to inefficient cooling (e.g., servers in the top of racks) and the service time of a server running at 50% of its maximum frequency is almost double that of a server running at its maximum frequency of operation (ignoring memory access and other delays). The relationship between the power consumption and thermal limitations of a server were derived in Chapter 4 and is given by Equation 4.1. Power budgets are allocated to servers so that the thermal constraints of individual servers are not violated and the capacity of servers are proportional to the power budgets.

Figure 5.3 shows the design of the load dispatcher. The load dispatcher in each tier accounts for the differences in capacities of the servers and adjusts the input to each server in the tier. The load dispatcher periodically infers the average service times of the individual servers and the current arrival rates for each server. The average utilization for tier i is then given by $\rho^i = \lambda^i/\mu^i$ where λ and μ are the average arrival rate and service times of the tier. The dispatcher in each tier i then adjusts the arrival rate of queries λ_j^i, for each server j so that the utilization $\rho_j^i = \lambda_j^i/\mu_j^i$ is the same ($= \rho^i$) for all servers j. The service rate reduction can be due to a number of reasons. The servers might be running at lower operational frequencies by means of techniques like *dynamic voltage and frequency scaling (DVFS)* [28]. An alternative technique is to force the servers to go to deep sleep states with low power consumption periodically in order to reduce their power consumption or prevent their temperatures from increasing beyond certain limits [5]. Irrespective of the control mechanism used, the load dispatcher has to make sure that the the load is shared proportional to the capacities (service rates) of the servers. During each integral control period ΔT_I the utilization of the tier is measured. The integral controller shown in Figure 5.3 periodically adjusts the weights for each server depending on the difference between the target ($\rho_i = \lambda_i/\mu_i$) and measured utilizations. The input to the controller is the measured utilization of each server. The error term $E(t)$ is the difference between the target and measured utilizations at time t. The new weights for the time period $(t, t+1)$ is given by the following equation.

$$Weight(t+1) = Weight(t) + K_i E(t) \tag{5.1}$$

where K_i is the integral constant. The arrival rates of each server for that period is then adjusted proportional to the weights assigned to each server.

5.2 Adapting to power constraints

With renewable energy resources and in some cases even with conventional power grids, the available power budget to the data center is not constant. In the multi–tier scenario that is discussed in this chapter, the primary adaptation mechanism that is used is controlling the number of servers that are powered on for serving the requests and the redistribution of requests to different servers. The former step is executed at time intervals of ΔT_S and the latter step is more frequent and is executed at time intervals of ΔT_I ($< \Delta T_S$). Given the available (predicted) power for the next ΔT_S period, the necessary control actions are initiated from the top–down.

Given the available power, the first step is to determine the number of servers that need to be powered on in each tier. The assumption to begin with is that there is an accurate estimate of the delays (response time) given the number of servers in each tier and their service rates. Let the number of tiers be M. As mentioned before, different servers have different thermal capacities and hence have different service rates.

Assuming that there are $N_1, N_2, N_3 ... N_M$ servers in each tier, if the capacity (constrained by thermal limits) of each server is accounted for individually and then the decision is made on which servers should be kept powered on in each tier to meet the overall delay bound, the corresponding algorithm would have a complexity of $O\left(2^{N_1+N_2+N_3...+N_M}\right)$. In order to avoid this, the following approximation scheme is used.

Step 1: At the beginning of each time interval ΔT_S, assume that all servers can be run at full capacity ignoring their thermal limits. Determine the minimum number of servers that need to be kept up and running in each tier to meet the overall delay bound - $O\left(N_1 N_2 N_3 .. N_M\right)$ algorithm. It is to be noted that typically $M \leq 3$ in most datacenters.

Step 2: Let the total number of servers in each tier i that need to be powered on given by the above step be N_i. Due to thermal constraints N_i servers that can operate at full capacity may not be available in a tier i. Hence the best servers are chosen so that the impact on delay is minimized .

The situation in Step 2 can be formulated as a knapsack problem. The knapsack problem can be defined as follows.

Definition 1 (Knapsack problem) : Given a knapsack of fixed capacity (here, $\sum_{i=1}^{i=N_i} P\left(S_i\right)$ where $P\left(S_i\right)$ is the power consumed by Server S_i) the objective is to fill the knapsack with a set of items ($\{I_k\}$) so that the total value of items ($\sum value\left(I_k\right)$, $\forall k$) packed in the knapsack is maximized and the sum of weights of the items ($\sum weight\left(I_k\right)$, $\forall k$) does not exceed the capacity of the knapsack.

Here the weight of each item ($weight\left(I_k\right)$) is the power consumed by the individual server and the total capacity of the knapsack is the total power budget of the tier. The assignment of the value of each item (server) ($value\left(I_k\right)$) is explained later. Knapsack problem is an NP–hard problem. A greedy approximate solution is used–the items are arranged in sorted order of $value(I_k)/weight(I_k)$. Then the items are picked one by one and packed into the knapsack as long as the sum of weights of the knapsack is less than the capacity of the knapsack. It is to be noted that an instance of the knapsack problem is solved at each tier and the capacity of the knapsack in each instance is the power budget of the tier. The total available power budget of the data center is allocated to each tier proportional to the number of servers that need to be kept powered on in that tier as determined in Step 1 above.

5.2.1 Assigning values to servers

This section explains how the *value* (I_k) is assigned to each server to include other factors besides service time. The objective of the knapsack algorithm is to maximize the value of items that are packed in the knapsack. It might take several seconds for a server to be powered up from a deep sleep/inactive state. This is called the activation period (T_a) of the server. Hence when choosing between multiple servers, the servers that are already up and running need to be preferred to the ones that are shut down or in deep sleep states. On the other hand when a server needs to be powered down, its workload needs to be redistributed to other servers which will in turn increase the load in the other servers. The pending sessions that the server was handling need to be completed. Above all the service rate of the server is the most important factor in determining its value. Hence the value function of each server needs to incorporate all these factors. Hence the value of sever I_k is given by the following equation.

$$value(I_k) = w_1\mu_{I_k} + w_2\lambda_{I_k} + w_3 R_{I_k} \tag{5.2}$$

where μ_{I_k} is the service rate and λ_{I_k} is the arrival rate of server I_k. It is to be noted that for a server that needs to be brought up and that was not handling any workload before, $\lambda_{I_k} = 0$. $R_{I_k} = 0$ if the server I_k is inactive and $R_{I_k} = 1$ if server I_k is already on. The servers with maximum value are packed into the knapsack i.e., kept powered on. Once the required number of servers are turned on, the load dispatcher that operates at a finer time granularity redistributes the workload to the different servers as explained in Section 5.1.1.

5.2.2 Estimating Delays

An important step in determining the number of servers in each tier is to determine the overall delay across all tiers given the current workload demand of the data center. Queuing theoretic modeling of the datacenter is used to estimate the delays involved. Let the arrival rate of requests into the data center be λ. As mentioned before, the load dispatcher in each level i adjusts the arrival rate to each server j in level i, λ_j^i so that the utilization $\rho = \lambda^i_j/\mu^i_j$ is the same for all the servers. The requests pass through the same tier multiple times. The average number of sessions in progress handled by the datacenter is N.

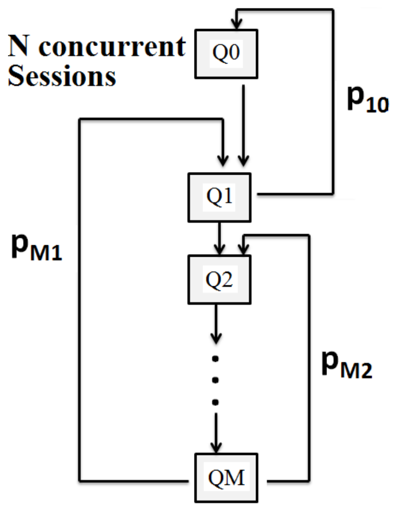


Figure 5.4: Closed Queue Model of a Datacenter

With the above assumptions, the multiple tiers can be analyzed as a closed network of queues [87, 95] as shown in Figure 5.4. The user requests originate from queue Q_0 and pass through each tier multiple times. The delay at Q_0 corresponds to the user think time which is the time spent by the user after receiving the response for a previous request and issuing a successive request. Q_0 is an infinite server queue that generates N concurrent sessions on the average. The traffic equation of each tier can hence be written as follows.

$$\lambda^j = \sum_{k=0}^{k=M} \lambda^k p_{kj}, j = 1, 2, ...N \tag{5.3}$$

where p_{kj} is the proportion of requests routed from queue k to queue j.

Mean Value Analysis (MVA) [87] is a popular technique for analyzing delays in networks of queues where each queue satisfies the $M \Rightarrow M$ property. This property states that if a queue is fed with Poisson input, the output process will also be Poisson. For a closed network, the MVA algorithm starts with the population of 1 and recursively computes the queue length and response time at higher population levels. The algorithm is shown in Algorithm 2.

Algorithm 2 MVA Algorithm

```
$n_i(0) = 0,\ i = 1, 2, 3\ ...\ \mathrm{M}$
$\tau_i(N) = \frac{1}{\mu_i} + \frac{1}{\mu_i} \times n_i\ (N-1)\ ,\ i = 1, 2, 3\ ...\ \mathrm{M}$
$\eta(N) = \frac{N}{\tau_0 + \sum_{i=1}^{i\,=\,M} \tau_i(N)}$
$n_i(N) = \eta(N)\tau_i(N)$ (Little's Law)
where, $\eta(N)$ is the throughput of the system with N customers
$\tau_i(N)$ is the average delay of the $i^{th}$ server when there are N customers in the system and
$n_i(N)$ is the average number of customers in queue $i$ when there are N customers in the system
```

In the case of FCFS scheduling discipline such as the one assumed in this work, $M \Rightarrow M$ property holds only for exponential arrival and service time distributions (M/M queues). Since exponential service time is not practical, a simple modification is used to extend the MVA for queuing networks where the $M \Rightarrow M$ does not hold as suggested in [87]. The N^{th} arriving customer will find $n_i(N-1)$ customers in service in tier i of which $U_i(N-1)$ will be busy already receiving service from the server in tier i. The average residual time of the customers is given by

$$\gamma_i = s_i \frac{(1 + CV_i^2)}{2} \tag{5.4}$$

where s_i is the service time of tier i and CV_i is the coefficient of variance of service times. The response time of the customer is therefore given by the following equation.

$$\tau_i(N) = \frac{1}{\mu_i}[1 + n_i(N-1)] + U_i(N-1)(\gamma_i - s_i) \tag{5.5}$$

Hence this value of $\tau_i(N)$ can be substituted in Algorithm 2 to get a more accurate estimate of the delay values for the FCFS service discipline with a general service time distribution. The mean service time of each tier depends on the thermal and power constraints of the individual servers and the arrival rates (assigned by the dispatcher). As before, the thermal constraints are ignored temporarily and it is assumed that each server in the tier can run at full capacity. Hence for different configurations i.e., different number of servers in each tier, MVA can be used to calculate the mean overall delay across all tiers for the requests. The configuration with the minimum power consumption that can satisfy the QoS guarantees for the requests is then chosen. For the chosen configuration, the knapsack problem instance is solved at each tier.

5.3 Planning

As explained in Section 5.2, at each time step ΔT_S the algorithm to optimize for power and performance based on the predicted power supply is executed and the number of servers that need to be powered on is adjusted. New servers need to be powered on and some servers need to be turned off. In either case, workload needs to be reassigned and the sessions that are already in progress need to be completed before the servers handling those sessions are turned off. All these operations cannot be done instantaneously and each of them has associated costs.

Let the initial state of servers at time t be $X(t)$ and the goal state given by the knapsack algorithm is $Y(t+T)$. Here the term state refers to the set of servers that are powered on and turned off and their workloads. The interval between time t and $t+T$ of length T is the planning horizon. A transition from state $X(t)$ to state $Y(t+T)$ involves multiple steps. An exploration of each possible state transition has a time complexity factor that is exponential in the number of servers involved. Hence we employ search heuristics that are well known and widely used in Artificial Intelligence for planning. One such technique to find the least cost path from the initial state to the goal state is the best first search algorithm [96]. The cost function $f(x)$ from one state to another represents the time taken to transition between the states. The next state that is chosen is the one with the minimum cost function $f(x)$.

Figure 5.5 shows the different state transitions during the execution of the required changes for adaptation. The following steps are involved during state transition. Let $\{S_i\}$ be the set of servers that are currently serving the requests. Let $\{A_i\}$ be the set of servers that need to be shut down and $\{B_i\}$ be the set of servers that need to be powered on. When a server has to be turned off, no new requests are forwarded to the server and the sessions that it is currently serving need to be completed. Hence the workload of the server to be shut down will have to be redistributed to other servers

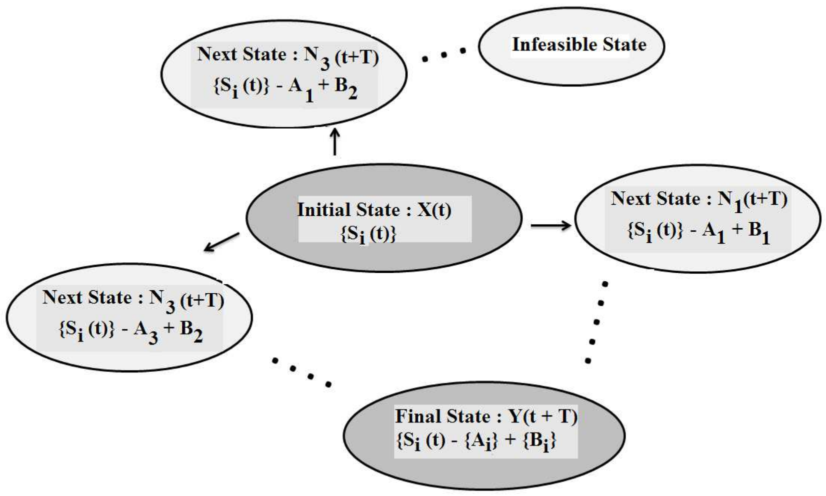


Figure 5.5: State transitions from initial to goal states

that are already up, thereby increasing the average delay in those servers. The path to reach the goal state from the start state needs to minimize the time taken for the control actions to be completed.

The cost function $f(x)$ which is the cost of getting to state x is therefore the estimated time to reach the state x from the start state. The cost function $f(x)$ is given by the following equation.

$$\begin{aligned} f(x) &= \max\left(\frac{Q_j(x)}{\mu_j(x)}, T_a\right), \;\; if\, N_A(x) \neq 0 \\ &= \frac{Q_j(x)}{\mu_j(x)}, \;\; if\, N_A(x) = 0 \end{aligned} \tag{5.6}$$

where j is the server with the maximum queue length in state x and $Q_j(x)$ is the queue length of j. $N_D(x)$ is the cardinality of the set of servers that are deactivated in state x. T_a is the time of activation of a server. $Q_j(x)/\mu_j(x)$ is the *virtual work* that needs to be completed before the server can be completely shut down. Any state is considered infeasible if the power consumption during or at the end of the state is more than the power budget of the tier. A server with a long queue length, will most likely have a large workload to be redistributed and hence shutting it down will increase the utilization of the other servers. Hence it is better to mark it for deactivation towards the end of the planning horizon since most of the servers that are marked for activation would be fully functional by then and will be ready to accept the workload of the servers with long queues.

It is to be noted that the constraints that are introduced for the feasibility of a state are along the power dimension. Orthogonally, strict constraints can also be introduced along the time dimension and the power consumed can be minimized. The planning process can also be made more dynamic by means of Model Predictive Control (MPC) [74] techniques where at time t, the actions are planned for every time instant $t + \tau$, $\forall \tau \in \{0, 1, ..T\}$ and only the control action for time $t + 1$ is implemented. Then at time $t + 1$ the same process is repeated for the receding time horizon T. However one drawback of this technique might be that this would turn out to be an expensive process considering the large state space. Besides these techniques might take a long time to converge and are not essentially optimal.

5.4 Evaluation

5.4.1 Simulator

This section evaluates the techniques presented in previous sections via detailed simulations. A discrete event simulator written in Java was used for evaluating the proposed scheme. The event queue is a priority queue ordered by event times. Each scheduled event has an associated *handler* class which executes the corresponding operations and schedules the next event when applicable. For example, the *Begin Processing* event assigns the request to a processor based on the weights given by the integral controller and then schedules an *End Processing* event after the service time for the request. As soon as a query arrives, it is sent to the wait queue of the tier. Based on the weights determined by the integral controller, the queries are assigned to appropriate processors. Then an *End Processing* event is scheduled. When the query has completed processing, it is either sent to the previous tier or to the next tier. In the simulations the data center has three tiers (besides the infinite server queue Q_0) and each tier processes a query twice except for the last tier. This models a datacenter as explained in Section 5.1 with a front–end tier, an application tier and a database tier. The integral controller in the dispatcher is invoked periodically at intervals of ΔT_I and whenever there is a change in the number of nodes in the tier. There is one integral controller object for each tier. The planning events occur at time intervals of ΔT_S. The total power supply varies every ΔT_S and the number of servers in each tier is determined by MVA. Then the knapsack problem instance is solved for each tier to determine which servers need to be powered on and which servers need to be shut down. The best first search algorithm is run to determine the best search path and then the steps in the path are executed one by one.

5.4.2 Simulation Parameters

Table 5.1: Power consumption at different CPU utilizations

Utilization	Performance Governor	Powersave governor
20%	$266W$	$266W$
40%	$327W$	$310W$
60%	$355W$	$340W$
80%	$385W$	$360W$
100%	$405W$	$380W$

The parameters in the simulator were set by experiments conducted on a Dell 2650 server with Intel Xeon processor and running Ubuntu Server. Linux kernel supports five different *governors* for scaling CPU frequency. The *performance governor* runs the CPU at the highest possible frequency and the *powersave governor* runs the CPU at the lowest possible frequency. Table 1 shows the power consumed by the server at different CPU utilization levels under the two governor settings. It can be seen that at lower utilization levels the power consumed is not significantly different for the two governor settings but at high utilization levels the difference in power consumed is high. It is to be noted that the server used in the experiments is an older server and hence the difference in power at different CPU frequency settings is not significant. In newer models, with a wider range of CPU frequencies, the difference in power consumption of the highest and the lowest frequency values might be even larger. The values of parameters c_1 and c_2 in Equation 4.1 were found to be 0.2 and -0.008 respectively as explained in Section 4.5.1 and the same values are used in the simulations.

As mentioned before, it is assumed that multiple commodity servers are used in each tier. It is assumed that the maximum number of servers in each tier is 40. There are 3 tiers of servers and each query passes through the first two tiers twice and the last tier once. The average service times of tier 1, 2 and 3 were assumed to be 20 ms, 40 and 120 ms respectively. The last tier typically interacts with slower storage components and hence the delay is higher. The overall processing time of a query that arrives into the system without incurring any waiting is $2 \times 20 + 2 \times 40 + 120 = 240$ ms. The utilization of the datacenter is changed by changing the number of concurrent sessions. It is assumed that the SLA requirements of the queries is that the average delay of the queries is within 1 sec. The power supply is assumed to consist of a combination of a renewable energy resource whose output varies periodically and a constant backup power from a conventional grid supply.

5.4.3 Simulation Results

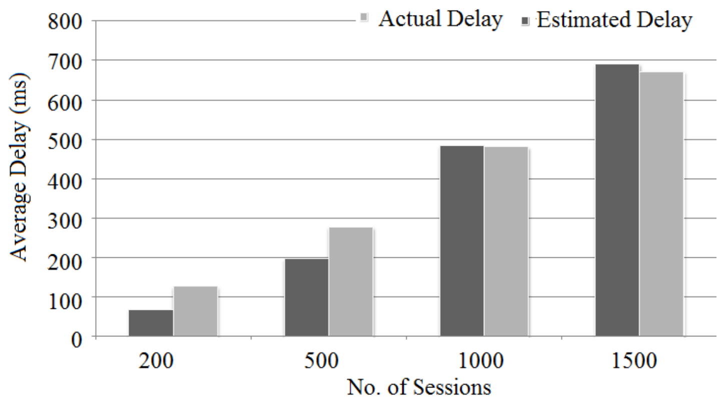
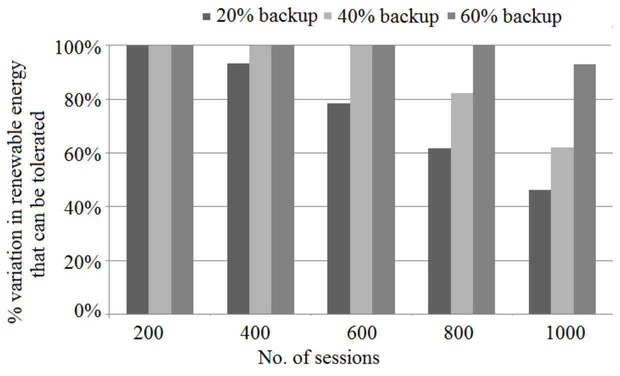
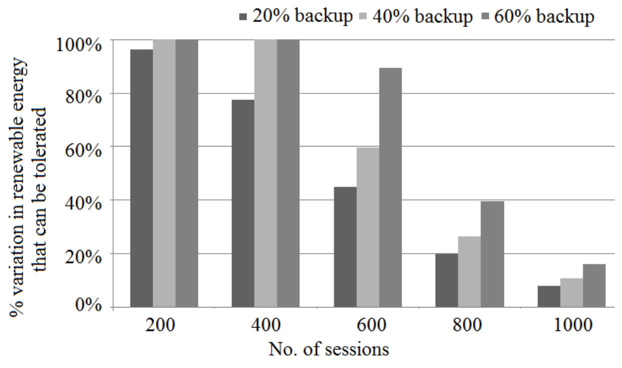


Figure 5.6: Actual delays vs estimated delays given by MVA

Figure 5.6 compares the actual average delay values and the average delay values as estimated by Mean Value Analysis when all servers are turned on. It is to be noted that the delay values given by MVA are an estimate of the average values and hence are not accurate. The estimated delay values are used as a guideline to determine the number of servers to be powered on in each tier. On the average the estimated delay values were within 15% of the actual delay values. To get more accurate estimates in real cases, a correction factor can be introduced by system profiling experiments to account for the estimation errors.



(a)



(b)

Figure 5.7: % variation in renewable energy that can be tolerated for different proportions of constant power (a) Tier power allocation based on MVA (b) Equal power allocation to all tiers

As explained before the power supply consists of a constant source of power (henceforth referred to as *brown energy*/backup power) and a renewable (*green*) energy resource. Together, these two resources can support the datacenter operations at full load ($\approx 50\,kW$). This model is different from the one assumed in Chapter 4 where the datacenters were powered by just the renewable energy source. The reason behind this model is that given the current landscape of renewable energy generation, renewable energy resources alone cannot support datacenter operations entirely [61]. Hence when all the servers in the datacenter operate at full load, a portion of the necessary power to support the operation of the datacenter is assumed to come from conventional (backup) power resources. The available backup power is denoted as BP kW. The remaining power comes from a renewable energy source and is denoted as GP kW. Hence when the renewable energy source is operating at full capacity, $BP + GP \approx 50$ kW. The simulations assume that all of the available green energy is utilized first before using any power from the backup power resources so that the environmental impact of *brown* energy consumption is minimized.

The following results analyze the energy profiles of renewable resources that the proposed scheme can successfully manage to adapt to. The amount of the available backup power is changed and the extent of variations in renewable energy that can be tolerated (i.e., the SLA requirement, delay $\leq 1000\,\text{ms}$, is not violated) is studied. The baseline scheme that is used to compare the efficiency of the proposed scheme is one where the available power is allocated equally to all tiers. This baseline scheme demonstrates the effect of failing to account for the difference in delays in different tiers when allocating power budgets. With the baseline scheme, when there are 1000 concurrent sessions, more than 100 servers need to be powered on to meet the SLA requirement. Figure 5.7 (a) shows the level of variations that can be tolerated with the proposed scheme. Let $TP = GP + BP$ be the total power required to support all servers in the data center. It can be seen that when the number of sessions is less (≤ 400), the proposed scheme can successfully maintain the delay bounds even when the renewable energy varies by as high as 90% (i.e, the renewable energy generated is between 0.1GP kW and GP kW) when the backup power is 20% of TP. Even when the number of sessions is as high as 1000, the tolerable limit for variation in renewable energy is almost 47%. However in the case of the baseline scheme as shown in Figure 5.7 (b), the tolerable variation limit is only 8% with a 20% backup power. This is because of the fact that the baseline scheme requires a lot more servers to be powered on than the MVA based allocation scheme in order to meet the delay bounds. It can be seen that as the proportion of backup power increases, the tolerable variation limit also increases as more power can be drawn from brown energy sources.

Figure 5.8 shows the proportion of green and brown energy required to support the datacenter operations when the available backup power is 20% of the total power required to support the entire datacenter ($0.2 \times 50 = 10$ kW in this case) and the renewable energy varies uniformly between 50% and 100% of the remaining 40 kW. It

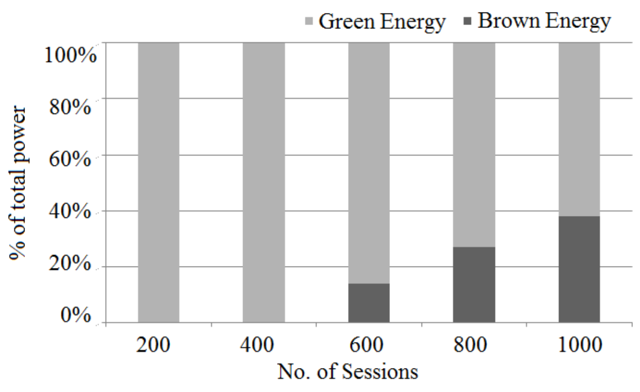


Figure 5.8: Proportion of green and brown energy consumed when the renewable energy varies between 50% to 100%

can be seen that when the number of sessions is small, the entire data center operations can be handled by renewable energy alone. When the number of sessions increases, the variability in renewable energy requires that some backup power needs to be used occasionally. It can be seen that even when the number of sessions is as high as 1000, only 38% of backup power is needed to guarantee the delay bounds. On the average, only 15% of the total power consumption comes from the brown energy source. This is because at each time instant in the proposed scheme only the required number of servers are kept powered on in each tier to meet the SLA requirements. It is to be noted that the consumption of *brown* energy is not explicitly optimized. The proposed scheme naturally reduces the energy consumption by powering on only the necessary number of servers in each tier to maintain the delay bounds.

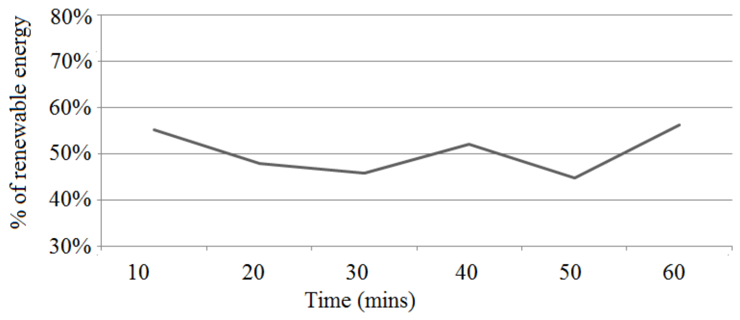


Figure 5.9: Power supply variation assumed for simulation

The following results show the performance of the proposed scheme under an energy constrained situation where the available energy is (at times) not sufficient to support the required number of servers to meet the SLA requirements. In order to simulate an energy constrained scenario, the following power supply model is assumed. A period of one hour where the available green energy fluctuates between 0.45 GP and 0.55 GP kW which corresponds to a variation level of 55%. It is to be noted that if the variation in available green energy is $\leq 48\%$, then according to Figure 5.7(a) there are no energy constraints and the delay bounds can be met with. Figure 5.9 shows the available renewable power profile that is assumed for the simulation and the associated variations. It can be seen that the available renewable energy fluctuates every 10 minutes. With the presence of adequate energy storage infrastructure, these fluctuations may be at the granularity of hours.

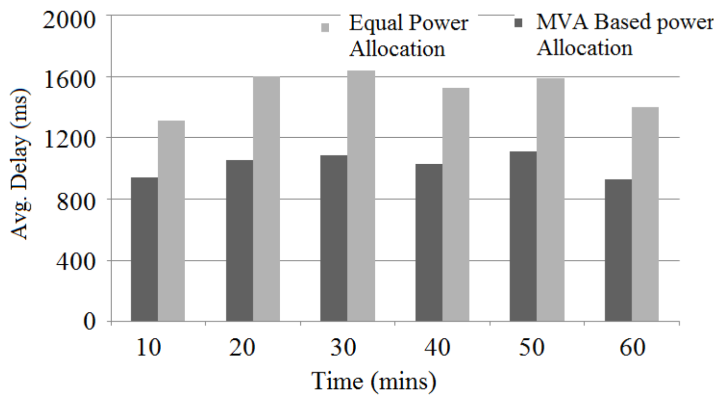


Figure 5.10: Average delays with 20% backup power and 1000 sessions

Figure 5.10 shows the delay incurred when the number of sessions is 1000 and the backup power is 20%. The power profile assumed is the same as shown in Figure 5.9. The baseline scheme that is used here is one where the available power budget is equally shared among all tiers. It can be seen that when the power budget is allocated equally, the delay values are 48% more than the proposed scheme on the average. A blind allocation of power to the tiers without acccounting for the tier delays will result in high waiting times even when a large number of servers are powered on.

A key contribution of this paper is the planning strategy for executing the control actions. Once the number of servers in each tier is determined for the next time period

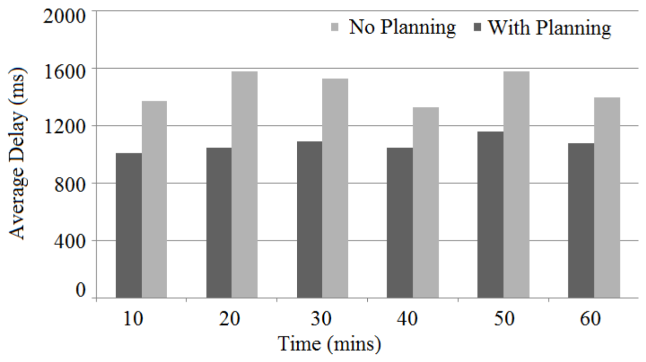


Figure 5.11: Average delays of queries during the control actions

by the knapsack algorithm based on MVA, the control actions for each step are given by the best first search algorithm. These include turning on and off the servers and redistributing the workloads. Figure 5.11 shows the delays of queries incurred during the control actions with 1000 concurrent sessions and the power profile as shown in Figure 5.9. The baseline scheme that is used to compare is one where the control actions (server activation, server deactivation, workload redistribution etc.) are initiated all at once as against the proposed proposed planned steps. It can be seen that the average delay of queries during the control period is reduced by around 26% compared to the baseline scheme. It is to be noted that the control period is shorter in the baseline scheme since it executes all actions at once. The benefit of the planning scheme is especially realized most when there is a drop in power budget as in the case of $t = 20$ and 50 minutes where the power budget drops by almost $3\,kW$. The baseline scheme allows for tranisient increase in power budgets since if all actions are done at once, the actual power budget may not be sufficient to support all operations.

Another important aspect of the proposed scheme is its ability to account for the thermal capacities of the nodes. This is realized by means of the load dispatcher. The load dispatcher takes the service rate of each node in the tier as input and allocates the appropriate arrival rates to the nodes so that the utilization is the same for all nodes. To demonstrate the effectiveness of the load dispatcher, the ambient temperatures of 50% of the nodes in each tier was set at $35°C$ and the rest of the nodes were at an ambient temperature of $25°C$. The nodes in the high temperature zones cannot operate at full capacity and their power consumption is limited by running them at a lower frequency

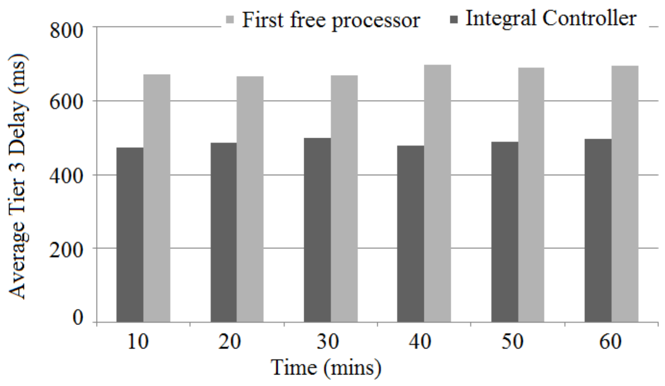


Figure 5.12: Average delay in nodes at high temperature zones

as given by Table 1. The maximum temperature limit was assumed to be $70°C$ and the power limit is determined by Equation 4.1. It is assumed that these servers are running at a lower frequency such that the service times are increased by 1.5 times the original service times. With these settings, the average delay values in the high temperature servers in Tier 3 is shown in Figure 5.12 for 1000 concurrent sessions. The baseline scheme that is used here for comparison is one that dispatches the queries to the first available free processor without considering the thermal limits/service rates of the nodes. It can be seen that the integral control based load dispatching scheme reduces the delay values by almost 40% compared to the baseline scheme. At low utilization levels, the delay values in these nodes is zero since they are not turned on. This is because when the knapsack problem instance is solved, these nodes are the least preferred to be packed into the knapsack because of low service rates. But at high utilizations, choosing these nodes becomes inevitable. However, the integral controller continuously adjusts the arrival rate into these nodes so that their average utilization is not high.

Chapter 6

Energy Adaptive Object Storage Framework

Scale–out storage architectures [97, 98] provide scalable storage that is accessible from all nodes in the datacenter. Data is typically stored as fixed or variable sized objects (or blobs) and each object has a unique identifier. This chapter provides the design and implementation of such a scalable object storage framework that adaptively provides the required performance and can gracefully degrade the performance in the presence of power constraints. The object storage framework is evaluated in the context of deduplicated virtual machine disks.

Virtualization of servers is becoming pervasive in datacenters due to a variety of reasons which include but are not limited to the isolation capabilities that it provides, efficient resource management, reduction in cost, power savings and ease of management. While virtualization of servers decreases the server related costs, the corresponding storage costs continue to increase since each of the virtual machines hosted in the physical servers still have their own individual disk files.

Deduplication of primary storage[1] disks [99, 100, 101] across these virtual machines reduces the storage space required to store these virtual disks and helps to manage the explosive growth of data in an affordable manner. Deduplication is the process of chunking (fixed sized or variable sized, typically 4 KB to 64 KB in size) the datasets and storing only the unique chunks on disk. The chunks are hashed and the hash values and locations are stored in an index. All redundant chunks occurring in the system point to the location of the unique chunks that have already been stored. In a *virtualized enterprise* environment where most of the virtual machines run the same operating systems and application binaries, the redundancy in virtual machine disks can be as high as 80% [99]. In the case of datacenters where typically there are several tera–bytes of backup storage, it has been shown that disk based deduplication can help to achieve a storage reduction ratio of up to *20 : 1* (Data size : Actual Storage) [102]. However unlike backup storage workloads, primary storage workloads are latency sensitive and any additional overhead that is caused by the process of deduplication could have an adverse impact on latency. When multiple VMs access the deduplicated disks from the shared storage, it generally leads to degradation in IO performance especially if many of the VMs are IO intensive. While the sever densities of racks and datacenters have

[1] The storage that is directly attached to the physical/virtual server is referred to as primary storage and any back–up storage is called secondary storage.

continued to increase, major infrastructure reconfiguration is needed to increase the peak power capacities of the racks and datacenters. Research efforts to use green/renewable energy sources to support most or all of the datacenter operations have to look at all consumers of power in a datacenter in order to be comprehensive. Due to the increased research efforts in the servers, the power consumed by servers has been continuously falling [52]. Serious efforts for power control in other areas like the networking and the storage subsystems are needed for a coordinated and comprehensive power management of the datacenter.

It follows from the aforementioned discussions that the storage infrastructure supporting datacenter workloads needs to be flexible, adaptive and scalable in order to meet with the performance and power constraints of the applications running in the datacenter. Storage systems constitute 40% of the overall datacenter power consumption [103]. As mentioned before, deduplication reduces storage capacity requirements considerably. In terms of power, this translates into a proportional storage power savings. The previous chapters investigated adaptations required in CPUs and networking components to stay within the power limits. The focus of this chapter energy adaptation of the storage infrastructure and this is done this via adaptive deduplication and replication. In particular, the design of a distributed storage framework called *flexStore*, that adaptively manages the deduplicated virtual machine disks is presented in detail. In short, three typical aspects of storage systems–deduplication, replication and inter–replica consistency are exploited and an adaptive framework that can meet with the performance and power constraints is developed and implemented.

6.1 Overview of *flexStore* Design Factors

6.1.1 Current Trends in Deduplication of VM Disks

While deduplication of backup data–streams has been studied for quite some time [102, 104, 105], deduplication of virtual machine disks (primary storage) has gained interest in the recent past [99, 101]. In virtualized environments, multiple virtual machines are hosted on a single physical machine. A hypervisor (e.g. Xen [106], VMWare ESX Server [88] etc.) moderates the allocation of hardware resources to the VMs in a transparent manner. All the (virtual) disk files of VMs are typically stored (with the extension

.vmdk in VMWare and *.img* in Xen) in the underlying storage on a shared storage pool and accessed by means of a cluster file system [99]. The shared storage architecture provides opportunity to perform deduplication on these virtual disk files, thus leading to a reduction in storage space. Especially in a private cloud environment where most of the operating systems and application files are similar in all of the virtual machines, the duplication might be large. It is also possible that the user data of the individual VMs like emails and code bases have a high degree of redundancy. Even in public clouds, there could be significant redundancy in virtual machine disks since most of the VMs are cloned from predefined templates. Jin et al. [100] provide a quantification of storage space savings in the VM disk images. Their findings indicate that there can be 70-80% savings in storage space for homogeneous VMs with almost similar disk images. Even in completely heterogeneous VMs they observe upto 40% savings in storage space. VMWare ESX server [107] implements an unobtrusive content based sharing of memory with COW (Copy On Write) semantics incorporated in it. Clements et al. [99] propose a decentralized deduplication technique for use in consolidated storage systems using Storage Area Networks (SAN). The authors have implemented the deduplication system on a cluster file system (VMFS) and report significant reduction in storage space in a virtualized enterprise environment. Nath et al. [108] analyze the use of content addressable storage in VM disk images in the presence of real time workloads from various users. 50-70% savings in networking resources. Most of the current works in primary storage deduplication have proposed doing the deduplication either on the individual host file systems [101, 109] or on a clustered file system [99]. The file system performs the deduplication and also manages the chunks on the storage system. This approach is highly inflexible for the following two reasons.

1. First, when multiple hosts are sharing the same storage system and each host manages its own chunks in its respective volume, there is no way to exploit the deduplication savings across multiple hosts' data.

2. Second, even in a setting where the clustered file system manages the chunks from multiple hosts, the management of chunk and file metadata makes the file system design extremely complicated and the system is limited by the scale of the file system. Any attempt to introduce the needed flexibility is very difficult

since it would require highly centralized file system components extended to more volumes.

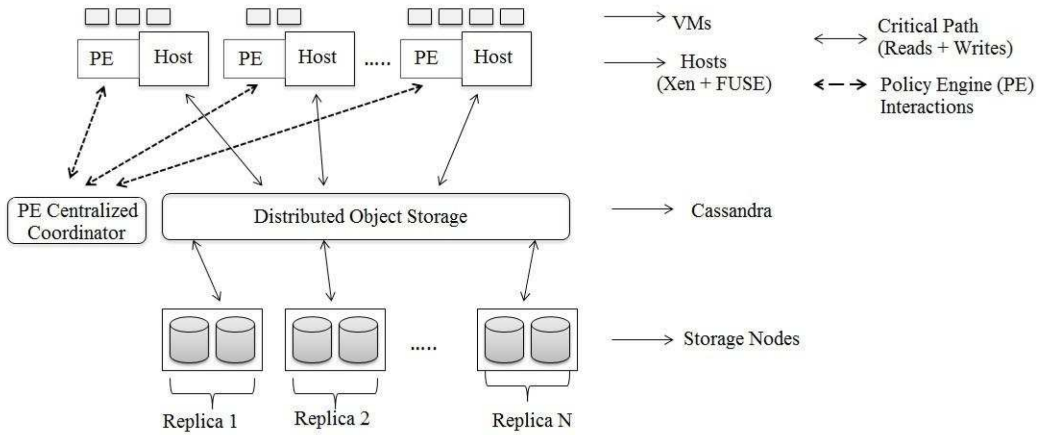

PE
Host
PE
Host
.....
PE
Host
VMs
Hosts (Xen + FUSE)
Critical Path (Reads + Writes)
Policy Engine (PE) Interactions
PE Centralized Coordinator
Distributed Object Storage
Cassandra
.....
Storage Nodes
Replica 1
Replica 2
Replica N

Figure 6.1: Design of *flexStore*

Replication has been traditionally used in storage systems to provide improved resilience and load balancing (for read workloads) (e.g. [60, 110]. In *flexStore* replication is used as a means to provide the required energy adaptation. *flexStore* separates out chunking and chunk management processes and these functions are carried out by a file system component and the storage system respectively. This demarcation of functionalities provides room for the required flexibility so that the storage system can dynamically adapt to the changes in workload and power constraints with minimum impact on performance.

6.1.2 Design of *flexStore*

The target environment consists of a number of physical hosts each running multiple virtual machines. There are three design components in *flexStore*– a file system component, a storage system component and a policy engine. Each VM stores its disk as a file in the local file system of the host. *flexStore* has a file system (FUSE [111]) module running in each host that does chunking of the VM disk files and maintains metadata only for the files in the respective host. The chunks are stored in the underlying distributed storage system (Cassandra [112]). The file system module has the necessary functions to access the chunks from the storage system and provides block level semantics to the VM disks. The next section describes the replication management in *flexStore* in detail. The storage system is a distributed object storage in which the objects are identified by the SHA1 hashes of their contents. The storage system maintains multiple replicas of the chunks. As mentioned before, *flexStore* does the adaptations by means of intelligently managing the replicas. Replica management in *flexStore* is detailed in Sections 6.1.3 and 6.1.4. A policy engine module that is present between the file system and storage system components provides the necessary interfaces to define the storage system policies and enforce the adaptation mechanisms. The policy engine also acts as an interface for the file–system module to pass any additional file–system level information like the I/O load of the individual VMs, VM user profile information etc. so that the storage system can store the chunks more intelligently. For instance, the storage system may attempt to store the chunks belonging to the same VM sequentially in the same order as they were written, since they are most likely to be read back in the same order. The policy engine in *flexStore* is designed as a distributed module and there are minimum centralized interactions and minimum interference with the regular read and write critical paths. The implementation of each of these modules is explained in detail in Section 6.2.

6.1.3 Replication Management

Unlike traditional storage systems that unify the storage in terms of logical volumes (LUNs) or NAS volumes, *flexStore* unifies the storage at the granularity of replicas. The distributed file system module specifically designed to talk to the storage system helps to provide the necessary block level abstractions. As mentioned before, *flexStore* does the adaptations by intelligently managing the replicas and dynamically adapting the assignment of VMs to replicas. Each replica has all the chunks needed by the deduplicated disks of all VMs it supports and may be spread over multiple storage nodes depending on the dataset size. A distributed policy engine module helps to define and enforce the required storage system policies. Figure 6.1 shows the various design components of *flexStore*.

6.1.4 Adaptive Replica–Consistency

Consider a datacenter with N replicas, R_1, R_2, R_3 .. R_N. Each VM is assigned to one replica and the replica serves both reads and writes for the VM. As new chunks are added to each replica, the replicas keep diverging continuously. It is important to minimize the divergence across the replicas for two reasons.

1. First, when adaptations are done, VMs may be reassigned to any replica. Let a VM be reassigned from R_2 to R_1. If R_1 already has all chunks of R_2, the reassignment of the VM will be much faster. Otherwise, either the chunks needed by the VM need to be copied over from R_2 to R_1 and only then the VM is reassigned or the VM is reassigned immediately to R_1 but still continues to read the chunks from R_2 until all chunks are copied over to R_1.

2. Second, if one replica fails, the chunks required by the VMs are available in other replicas and hence the storage system can be made more resilient to failures.

When new chunks are written, a log file records these chunks and then when the synchronization process starts, the logged chunks are copied over from one replica to others. The resources in the datacenter need to be shared between the regular I/O operations of the workloads and the I/O operations caused by synchronization. Any available resource in replica r, $I(r)$, hence has two parts- $I_1(r)$ and $I_2(r)$. I_1(r) is the amount of resource used by regular I/O operations in replica r and $I_2(r)$ is the amount of resources used by replica synchronization operations in replica r. Let $I_2^{thresh}(r)$ be the maximum amount of resources allocated to synchronization I/O operations. Hence the actual amount of resources available for regular I/O operations is given by,

$$I_1(r) = I(r) - I_2(r), \quad I_2\,(r) \leq I_2^{thresh}(r) \tag{6.1}$$

By dynamically adjusting the value of $I_2^{thresh}(r)$, the consistency level provided by the system can be adjusted. For instance, if $I_2^{thresh}(r)$ is large , replica synchronization can use up the maximum amount of resources and hence the system can provide close to synchronous replication at the cost of I/O performance. On the other hand, if $I_2^{thresh}(r)$ is small, the synchronization process does not get the required I/O resources during peak I/O demands.

Let the network bandwidth, disk bandwidth and power budget available for synchronization in replica R_i be $BW_{nw}^{avail.}(i)$, $BW_{disk}^{avail.}(i)$ and $PB^{avail.}(i)$ respectively. Let the network bandwidth, disk bandwidth and power budget used for synchronization between replica R_i and R_j be $BW_{nw}(i,j)$, $BW_{disk}(i,j)$ and $Power(i,j)$ respectively. To avoid redundancy in discussion of network and disk bandwidths, in general, any available bandwidth (network/disk) is denoted by $BW^{avail}(i,j)$ and the bandwidth used for synchronization is denoted by $BW(i,j)$. Let $D(i,j)$ denote the number of chunks that need to be synchronized between replica R_i and R_j (note that $D(i,j) \neq D(j,i)$). The problem of optimizing the process of synchronizing the replicas can be formulated as follows. Let $m_{i,j}$ be a boolean and the chunks are copied from replica R_i to R_j only if $m_{i,j} = 0$.
The objective function is,

$$minimize \sum_{i=1,j=1}^{i=N,j=N} D\left(i,j\right) * m_{i,j}\ , \quad \forall i,j \in \{1,2,..N\}, \quad m_{i,j} = 1\, or\ 0 \tag{6.2}$$

subject to the constraints,

$$BW(i) = \sum_{j=1}^{j=N} m_{i,j}^{c} BW(i,j) + \sum_{j=1}^{j=N} m_{j,i}^{c} BW(j,i) \leq BW^{avail.}(i)\ , i = 1,2,3...N \tag{6.3}$$

$$Power(i) = \sum_{j=1}^{j=N} m_{i,j}^{c} Power(i,j) + \sum_{j=1}^{j=N} m_{j,i}^{c} Power(j,i) \leq PB^{avail.}(i)\ , i = 1,2,3...N \tag{6.4}$$

where $m_{i,j}^{c}$, is the compliment of $m_{i,j}$.

Equation 6.3 is applicable to both disk and network bandwidths. The solution to the above optimization problem is a vector $[M_{i,j}]$ where the entries are either $m_{i,j}$ =1 or 0. The chunks are copied from replica R_i to replica R_j if $m_{i,j} = 0$. The above convex optimization problem is an Integer Linear Program (ILP) and the complexity of finding a solution is $O\left(2^N\right)$. Since the number of replicas in the storage system is very small, (typically N=2 or N=3), a simple branch and bound technique is used to solve the ILP.

The bandwidth constraints are enforced as follows. The required I/O bandwidth for synchronization from replica R_i to replica R_j depends on the number of chunks that need to be synchronized. Let the control period for which the synchronization decisions are made be T_s. Hence the bandwidth required is given by,

$$BW^{reqd}(i,j) = [D(i,j) * chunksize]/T_s \tag{6.5}$$

Hence the bandwidth used in Equation 6.3 is given by,

$$BW(i,j) = min(BW^{reqd}(i,j), BW^{remaining}(i), BW^{remaining}(j)), \tag{6.6}$$

where $BW^{remaining}(i)$ is the amount of available bandwidth at replica R_i at the current step in the solution to the problem in Equation 6.2.

Equation 6.5 and 6.6 also apply to both network and disk I/O bandwidth. A minimum bandwidth is reserved in all nodes for the synchronization process in order to avoid the synchronization process to be prevented for long periods due to starvation of resources. It is also assumed that when the synchronization process starts, even replicas that are shut down are brought up for a brief period and they receive the updates and then they are shut down back. Since these replicas are not serving any active reads or writes, the synchronization can use all the resources available in these replicas. However this requires additional power. This power is assumed to come from the back up (brown) power sources.

6.2 Implementation of Components in *flexStore*

As explained before, *flexStore* has a file system component and a storage system component. A policy engine module helps to define and enforce the various storage system policies.

6.2.1 File System Component

The file system component was implemented as a FUSE (Filesystem in User Space) module [111]. The VM disks are stored in the mountpoint where the FUSE module is mounted in the host. The reads and writes to the VM disk files are handled in the FUSE module which also manages the metadata for the VM disk files. The FUSE metadata structure is persistent. However, in order to measure the latencies involving just the storage system and eliminate any impact of metadata lookups in the host on latencies, sufficient memory was allocated in each host so that the entire metadata structure fits in memory. Besides, this is not an unreasonable assumption since the hosts typically have large portions of memory allocated for the file system operations and the disk accesses for index lookups are typically well optimized [113].

6.2.2 Distributed Object Store

Cassandra is used for storing the chunks. Cassandra is a distributed key value store and is highly scalable and provides good performance for single row accesses in a large dataset. Besides Cassandra has no single point of failure and each node can perform all functions including coordination of the cluster. Each Cassandra node has a unique token that decides what range of keys go to that particular node. By appending appropriate prefixes to the SHA1 keys, the chunks can be directed to specific nodes. Thus each replica is placed in a specific set of nodes. Whenever a VM sends a read request for a chunk with a particular SHA1 value, the appropriate replica prefix is appended to the SHA1 and thus the read requests are redirected to the appropriate Cassandra node. Each Cassandra node runs a thrift server to which the thrift client in the FUSE module talks to. The thrift server is responsible for fetching the chunks from the local Cassandra daemon and sending it back to the file system client. The thrift server in each replica also logs the new chunks which are later synchronized with other replicas.

6.2.3 Policy Engine

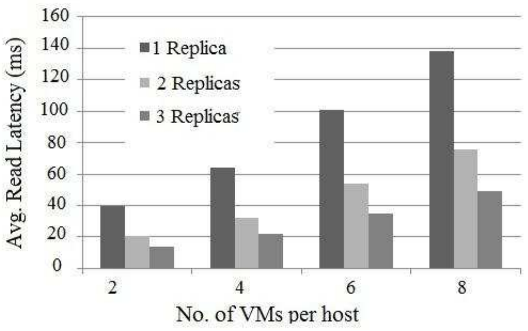
The distributed policy engine module in each host is a daemon that manages and receives periodic updates about VM to replica mapping, I/O load information of each of the VMs and the I/O load of the replicas. The policy engine module consists of individual instances that maintain an authoritative copy of the local state data. These instances periodically publish their local state to a central entity, that reconciles the state information and publishes the global state to all instances. The instances running on individual nodes interact with the FUSE modules to receive information about the I/O load per VM and also the total I/O load contribution of the host per replica, the instance specifies the VM to replica mapping to the filesystem (FUSE) module. These interactions between the application instances and the filesystem (FUSE)module are not on the critical path and happen only during state changes. The policy engine provides an interface to specify the different policy requirements like minimum number of replicas to keep synchronized at any time, the SLA requirements (e.g. average latencies) of the VMs, the consistency level across multiple replicas (e.g. strong, medium, weak) etc.

6.3 Evaluation

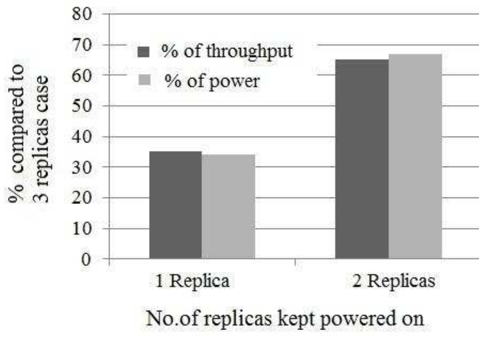
This section presents the experimental results on a cluster of servers.

6.3.1 Experimental Setup

The experimental testbed consists of 2 hosts and 3 storage nodes. Every node has 8 cores and 8 GB of memory and disks attached to individual JBOD boxes connected to each of the host. The nodes are interconnected by 1 Gbps ethernet. The operating system in each of the nodes was CentOS server 6.3 and virtual machines run on Xen in the two host nodes. The FUSE module and the policy engine run on the two hosts and the replication factor was set to 1 in Cassandra unless specified otherwise. Microbenchmarks (to control the amount of deduplicated data) and *fio* [114] (a standard benchmark to evaluate file system workloads) on the virtual disks were used to evaluate the properties of *flexStore*. The power profiles that were used were from the National Renewable Energy Laboratory (NREL) [115] wind power datasets. The power values were scaled according to the cluster size that is used in the experiments and the variations in power supply of the original dataset were retained. Also, it is assumed that there is sufficient power (either entirely green power or green power + backup brown power) to carry out the foreground and background operations at any point in time once the number of replicas that are kept powered on is determined. In all the experiments the chunk size was 4 KB which was the same as the OS page size in the VMs. In the presence of write workloads, the same rate of new chunk generation (via benchmarks) was used for all cases the performance is compared with other schemes.



(a)



(b)

Figure 6.2: (a) Average read latencies with different number of replicas. (b)Reduction in throughput and power compared to 3 replicas case when there are 8 VMs per host.

6.3.2 Experiments and Results

Figure 6.2(a) shows the latencies with different number of VMs running in each of the two hosts for one, two and three replicas and a zipf access distribution of blocks. We see that as the number of VMs increases, as expected, the latency also increases due to interference between multiple VMs. However, we see that when 3 replicas are serving the VMs, the average read latency decreases by around 70% irrespective of the number of VMs when compared to the case when there is only one replica serving the VM. These results are a proof of the basic premise of the *flexStore* design that increasing the number of replicas in the object storage indeed helps to improve the read performance in the chunk storage under various utilization levels. Figure 6.2(b) shows the percentage of I/O throughput achieved compared to the case when the number of replicas is 3 and the percentage of power consumed when the number of replicas is 3 (when there are 8 VMs running per host). Although it is important to consider both power and energy management, in the context of energy adaptive computing (and in this work), the crucial objective is to stay within the power limitations enforced by the varying renewable power supply and minimize brown power consumption. From Figure 6.2(b) it can be seen that the reduction in throughput is accompanied by an almost equal reduction in power consumed. In other words, adjusting the number of replicas provides a power proportional control which is important to stay within the power cap. This also means that there is no significant savings in total *energy* consumed but the system successfully stays within the power cap. We plan to consider the problem of limiting the total energy consumption of the system as future work.

Figure 6.3 shows the average IO latencies with 3 replicas, 8 VMs and different consistency levels. In the case of *strong consistency*, the writes return only after the data has been written to all 3 replicas while in the case of *weak consistency* and *flexStore*, the writes return immediately after being written to one replica. Hence the write latency is significantly higher in *strong consistency* than the other two cases. The VMs are assigned to replicas based on a greedy policy that balances the number of IOPS of all the replicas that are powered on. In the case of *weak consistency*, the replicas are synchronized only when a VM is reassigned from one replica to another. In the case of *flexStore* replica synchronization happens periodically. The write latencies are practically the same in *weak consistency* case and *flexStore* since in both the cases, the writes return immediately. Similarly, the impact on read latencies are comparable in both the cases but *flexStore* achieves this despite the presence of periodic synchronization. However it is to be noted the impact on latency depends on the frequency of synchronization and in *flexStore* we carefully choose the frequency of synchronization based on apriori knowledge of application characteristics via profiling.

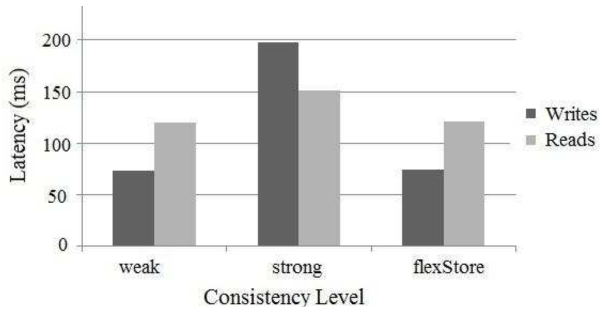


Figure 6.3: Average IO Latencies for different consistency levels with simultaneous reads and writes

Figure 6.4 shows the latency values and the power variations for an hour of experimental run. Initially there are three replicas serving the VMs. At time t=11 min (as shown by the first small circle), the available power goes down and only two replicas can be supported. The transition takes longer time in the case of *weak consistency* because all the new (unsynchronized) chunks from the replica in the node to be shut down have

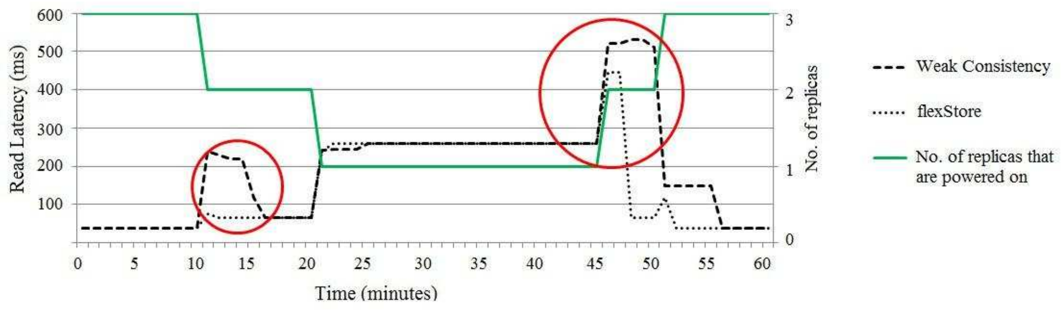


Figure 6.4: Comparison of impact of adaptation actions

to be moved over to the other replicas and only then the node can be shut down. This overhead is considerably lower in the case of *flexStore* since the replicas have already been partially synchronized and there are fewer chunks to be copied from the node that is being shut down and hence the merge is much faster. Similarly, in the opposite case when the available power increases and the number of replicas can be increased as in time t=45 min (shown in the bigger circle), the time taken for the latency values to reduce is higher in the case of *weak consistency* than *flexStore*. This is because of an interesting tradeoff in *flexStore*. As mentioned earlier, the power supply consists of variable (green) energy source and a constant backup (brown) power supply. In the case of *flexStore* even though the replica is down, when the synchronization process starts, the replicas that are shut down also get the updates. This means that they consume some power during the synchronization which is derived from the brown energy source. If bringing up the replicas just for synchronization process is not feasible, then in case there is an increase in the number of replicas, the performance overhead of *flexStore* is similar to that of *weak consistency* case. One of the key aspects of *flexStore* is this flexibility it provides in choosing different strategies to adapt to the constraints.

Figure 6.5 shows the time taken to bring a node down, with the same amount of new chunks being generated during steady state, in the case of *weak consistency*, *flexStore* and the default Cassandra replication strategy. As explained before, *flexStore* takes lesser time than *weak consistency* case since the replicas are periodically synchronized in the case of *flexStore*. In the case of Cassandra, by default, the first replica is placed on the node handling the key range and the other replicas are placed in the subsequent nodes in the ring. Hence with 3 nodes, it is fair to compare with the case of Cassandra

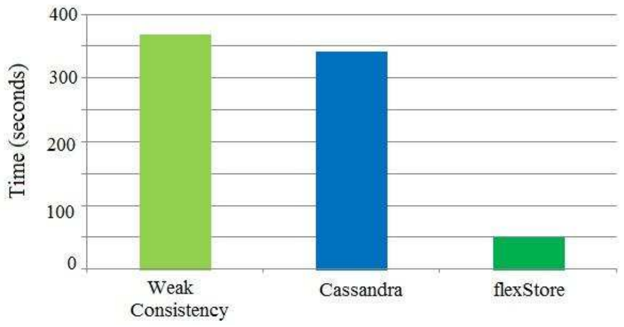


Figure 6.5: Time taken to reduce from 3 nodes to 2 nodes

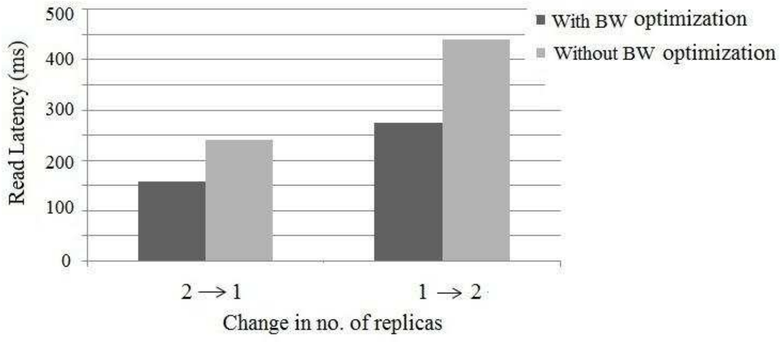


Figure 6.6: Effect of Bandwidth optimization in *flexStore*

with a replication factor of only 2 because there are chunks in one node that are not present in others. When a node is decommissioned, the chunks that are present in the decommissioned node are copied over to the other nodes. Since in Cassandra, the replicas are distributed to the nodes based on the hash space, it takes longer time to copy (one third of the overall new chunks generated) the chunks from the decommissioned node to the other nodes. This demonstrates the fact that controlling the placement of replicas is also important to reduce the overhead of adaptation operations.

Figure 6.6 shows the importance of optimization of resources used by the synchronization process in *flexStore*. The figure shows the read latencies observed when the number of replicas changes from 2 to 1 and also from 1 to 2. The bandwidth that can be used for synchronization to meet the latency requirements of the VMs was profiled and was used as $I_2^{thresh}(r)$ in Equation 6.1. It can be seen that the impact on latency of the foreground reads is lesser ($\approx 35\% - 37\%$) when the bandwidth available for background synchronization is adjusted dynamically than when the available bandwidth is simply shared between the foreground and background operations.

Chapter 7

Conclusion

In this thesis a new paradigm called Energy Adaptive Computing (EAC) was introduced and its realization in datacenters was explored under different scenarios and workloads. The various chapters discussed how EAC can make IT more sustainable and elaborated on the types of EAC scenarios and the challenges associated with each of them. A major goal of this work is to inspire right-sizing the otherwise over designed infrastructure in terms of power and ensure the possibility of addressing the ensuing challenges via smarter control. This thesis discussed the challenges involved in adapting to the energy and thermal profiles in a data center and proposed adaptation mechanisms without compromising on the QoS too much. EAC puts power/thermal controls at the heart of distributed computing.

Chapter 4 presented the design of *Willow*, a simple control scheme for energy and thermal adaptive computing. Chapter 5 investigated the control actions that need to be executed at different time granularities and their implementation in a multi–tiered datacenter. It has been shown that an efficient implementation of the control actions can help reduce the associated overheads significantly and adapt to significant variations in the available power supply. Chapter 6 presented the design, implementation and evaluation of *flexStore*, a storage framework that can adapt to changes in workload and available power. The experimental results show that *flexStore* provides opportunities to gracefully degrade the performance in the presence of energy constraints. The framework also manages replication adaptively so that both the performance and power constraints can be satisfied. *flexStore* provides various adaptation parameters in the policy engine that can be used to define different storage system policies like latency requirements, resilience etc.

Future Work:

A real time implementation might need to consider the migrations that are caused as a result of resource constraints as well. In order to do a holistic power control, the adaptation must consider the energy consumed by cooling infrastructure as well in the adaptation. Another interesting area that is worth exploring is the possibility of predicting the variations in energy profiles and doing some computations in advance in anticipation of energy shortages. An immediate extension to this work would be the case where the datacenters are geographically distributed clusters in which case the network (WAN) costs and network partitioning issues play a significant role in datacenter management. It would also be interesting to include the variations in electricity markets and adaptively schedule the jobs or route the requests to different geographically located datacenters. In this regard, the schemes proposed in works such as [116, 117] would be complementary to the schemes proposed in this thesis as the goal of this thesis would be to take the energy variations and current load in a datacenter as input and adapt to the variations in both simultaneously. As mentioned earlier, this thesis aims at adapting to the energy variations and hence does not focus too much on energy efficiency per se. Optimizing the energy consumed of the system explicitly in addition to the power adaptations is also a possible future extension of this work. An interesting area to be investigated is the impact of the storage system adaptations on the resource allocations on the host side. For instance if the number of replicas is less, the VMs spend more time waiting and hence the CPUs of the hosts remain idle for longer periods. This might provide additional opportunities to do power management in the hosts as well. A more complete control system for cluster EAC must be able to measure power consumption and temperature of every component in the server including memory, NIC, hard disks etc. and make fine grained control decisions. A future extension could consider more complex EAC scenarios including the problems of dividing up available energy between servers, storage and network such that the application progress can be optimized. The adaptation techniques for cluster EAC under more complex workloads where there is excessive IPC traffic among the servers in addition to requests from clients is also an interesting area to explore.

References

[1] Amazon.com Inc. `http://www.amazon.com`.

[2] eBay Inc. `http://www.ebay.com`.

[3] Facebook Inc. `http://www.facebook.com`.

[4] Twitter Inc. `http://www.twitter.com`.

[5] Navin Sharma, Sean Barker, David Irwin, and Prashant Shenoy. Blink: Managing server clusters on intermittent power. In *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems (ASPLOS)*, 2011.

[6] Google Green Energy Initiative. `http://www.google.com/about/datacenters/energy.html`.

[7] Apple and the environment. `http://www.apple.com/environment/renewable-energy/`.

[8] Akshat Verma, Pradipta De, Vijay Mann, Tapan Kumar Nayak, Amit Purohit, Gargi Dasgupta, and Ravi Kothari. BrownMap: Enforcing Power Budget in Shared Data Centers. In *Middleware*, 2010.

[9] K. Kant, M. Murugan and D. H. C. Du . Enhancing data center sustainability through energy adaptive computing. *ACM Journal on Emerging Technologies in Computing Systems (Special Issue)*, April 2012.

[10] Krishna Kant, Muthukumar Murugan, and David H.C.Du. Willow: A Control System for Energy and Thermal Adaptive Computing. In *Proceedings of 25th*

IEEE International Parallel & Distributed Processing Symposium, IPDPS '11, 2011.

[11] M.Murugan, K.Kant, and D. Du. Energy Adaptation for Multi-tiered Datacenter Applications. *Intel Technology Journal*, 16, 2012.

[12] Vasanth Venkatachalam and Michael Franz. Power reduction techniques for microprocessor systems. *ACM Comput. Surv.*, 37(3):195–237, 2005.

[13] Luiz André Barroso and Urs Hölzle. The case for energy-proportional computing. *IEEE Computer*, 40(12):33–37, 2007.

[14] Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In *NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, 2008.

[15] Canturk Isci, Alper Buyuktosunoglu, Chen-Yong Cher, Pradip Bose, and Margaret Martonosi. An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget. In *39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-39 2006), 9-13 December 2006, Orlando, Florida, USA*, 2006.

[16] Jeffrey S. Chase, Darrell C. Anderson, Prachi N. Thakar, Amin M. Vahdat, and Ronald P. Doyle. Managing energy and server resources in hosting centers. *SIGOPS Oper. Syst. Rev.*, 35(5):103–116, 2001.

[17] Chengmo Yang and Alex Orailoglu. Power efficient branch prediction through early identification of branch addresses. In *CASES '06: Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems*, 2006.

[18] Dharmesh Parikh, Kevin Skadron, Yan Zhang, and Mircea Stan. Power-Aware Branch Prediction: Characterization and Design. *IEEE Trans. Comput.*, 53(2):168–186, 2004.

[19] Krishna Kant. A control scheme for batching dram requests to improve power efficiency. In *SIGMETRICS*, pages 139–140, 2011.

[20] Krishna Kant. Power control of high speed network interconnects in data centers. In *INFOCOM'09: Proceedings of the 28th IEEE international conference on Computer Communications Workshops*, pages 145–150, 2009.

[21] Maruti Gupta and Suresh Singh. Dynamic Ethernet Link Shutdown for Energy Conservation on Ethernet Links. In *ICC*, pages 6156–6161, 2007.

[22] Sudhanva Gurumurthi, Anand Sivasubramaniam, Mahmut Kandemir, and Hubertus Franke. DRPM: dynamic speed control for power management in server class disks. *SIGARCH Comput. Archit. News*, 31(2):169–181, 2003.

[23] Dennis Colarelli and Dirk Grunwald. Massive arrays of idle disks for storage archives. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–11, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.

[24] B.Heller, S.Seetharaman, P.Mahadevan, Y.Yiakoumis, P.Sharma, S.Banerjee, and Nick McKeown. ElasticTree: Saving Energy in Data Center Networks. In *NSDI'10: Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation*, 2010.

[25] Omid Azizi, Aqeel Mahesri, Benjamin C. Lee, Sanjay J. Patel, and Mark Horowitz. Energy-performance tradeoffs in processor architecture and circuit design: a marginal cost analysis. In *ISCA*, pages 26–36, 2010.

[26] Yefu Wang, Kai Ma, and Xiaorui Wang. Temperature-constrained power control for chip multiprocessors with online model estimation. *SIGARCH Comput. Archit. News*, 37(3):314–324, 2009.

[27] Daniel Chaver, Miguel A. Rojas, Luis Pinuel, Manuel Prieto, Francisco Tirado, and Michael C. Huang. Energy-aware fetch mechanism: trace cache and BTB customization. In *ISLPED '05: Proceedings of the 2005 international symposium on Low power electronics and design*, pages 42–47, 2005.

[28] Gaurav Dhiman and Tajana Simunic Rosing. Dynamic voltage frequency scaling for multi-tasking systems using online learning. In *ISLPED '07: Proceedings of the 2007 international symposium on Low power electronics and design*, pages 207–212, 2007.

[29] Urs Hoelzle and Luiz Andre Barroso. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 1st edition, 2009.

[30] L. Benini, G. De Micheli, E. Macii, D. Sciuto, and C. Silvano. Address bus encoding techniques for system-level power optimization. In *Proceedings of the conference on Design, automation and test in Europe*, DATE '98, pages 861–867, Washington, DC, USA, 1998. IEEE Computer Society.

[31] Tiehan Lv, Jörg Henkel, Haris Lekatsas, and Wayne Wolf. A dictionary-based en/decoding scheme for low-power data buses. *IEEE Trans. Very Large Scale Integr. Syst.*, 11:943–951, October 2003.

[32] Mircea R. Stan and Wayne P. Burleson. Bus-invert coding for low-power i/o. *IEEE Trans. Very Large Scale Integr. Syst.*, 3:49–58, March 1995.

[33] Ke Ning and David R. Kaeli. Bus power estimation and power-efficient bus arbitration for system-on-a-chip embedded systems. In *PACS*, pages 95–106, 2004.

[34] Qingyuan Deng, David Meisner, Luiz Ramos, Thomas F. Wenisch, and Ricardo Bianchini. Memscale: active low-power modes for main memory. In *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems*, ASPLOS XVI, pages 225–238, New York, NY, USA, 2011. ACM.

[35] Hai Huang, Padmanabhan Pillai, and Kang G. Shin. Design and implementation of power-aware virtual memory. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '03, pages 5–5, Berkeley, CA, USA, 2003. USENIX Association.

[36] Parthasarathy Ranganathan, Phil Leech, David Irwin, and Jeffrey Chase. Ensemble-level power management for dense blade servers. In *Proceedings of the 33rd annual international symposium on Computer Architecture*, ISCA '06, pages 66–77, 2006.

[37] Justin Moore, Jeff Chase, Parthasarathy Ranganathan, and Ratnesh Sharma. Making scheduling "cool": temperature-aware workload placement in data centers. In *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 5–5, 2005.

[38] Arun Babu Nagarajan, Frank Mueller, Christian Engelmann, and Stephen L. Scott. Proactive fault tolerance for hpc with xen virtualization. In *Proceedings of the 21st annual international conference on Supercomputing*, ICS '07, pages 23–32, 2007.

[39] Akshat Verma, Puneet Ahuja, and Anindya Neogi. Power-aware dynamic placement of HPC applications. In *Proceedings of the 22nd annual international conference on Supercomputing*, ICS '08, 2008.

[40] Ripal Nathuji and Karsten Schwan. VirtualPower: Coordinated power management in virtualized enterprise systems. In *SOSP '07: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, pages 265–278, 2007.

[41] Ripal Nathuji and Karsten Schwan. Vpm tokens: virtual machine-aware power budgeting in datacenters. In *Proceedings of the 17th international symposium on High performance distributed computing*, HPDC '08, pages 119–128, New York, NY, USA, 2008. ACM.

[42] Xiaorui Wang and Yefu Wang. Coordinating Power Control and Performance Management for Virtualized Server Clusters. *IEEE Transactions on Parallel and Distributed Systems*, 99, 2010.

[43] 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 *Proceedings of the 5th USENIX*

Symposium on Networked Systems Design and Implementation, NSDI'08, pages 337–350, Berkeley, CA, USA, 2008. USENIX Association.

[44] David Meisner, Brian T. Gold, and Thomas F. Wenisch. Powernap: eliminating server idle power. In *Proceedings of the 14th international conference on Architectural support for programming languages and operating systems*, ASPLOS XIV, pages 205–216, New York, NY, USA, 2009. ACM.

[45] Ramya Raghavendra, Parthasarathy Ranganathan, Vanish Talwar, Zhikui Wang, and Xiaoyun Zhu. No "power" struggles: coordinated multi-level power management for the data center. In *Proceedings of the 13th international conference on Architectural support for programming languages and operating systems*, ASPLOS XIII, pages 48–59, New York, NY, USA, 2008. ACM.

[46] Priya Mahadevan, Puneet Sharma, Sujata Banerjee, and Parthasarathy Ranganathan. A power benchmarking framework for network devices. In *Proceedings of the 8th International IFIP-TC 6 Networking Conference*, NETWORKING '09, pages 795–808, 2009.

[47] Maruti Gupta and Suresh Singh. Greening of the internet. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 19–26, New York, NY, USA, 2003. ACM.

[48] Priya Mahadevan, Puneet Sharma, Sujata Banerjee, and Parthasarathy Ranganathan. Energy aware network operations. In *Proceedings of the 28th IEEE international conference on Computer Communications Workshops*, INFOCOM'09, pages 25–30, 2009.

[49] Ganesh Ananthanarayanan and Randy H. Katz. Greening the switch. In *Proceedings of the 2008 conference on Power aware computing and systems*, HotPower'08, pages 7–7, 2008.

[50] Yuvraj Agarwal, Steve Hodges, Ranveer Chandra, James Scott, Paramvir Bahl, and Rajesh Gupta. Somniloquy: augmenting network interfaces to reduce pc

energy usage. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, NSDI'09, pages 365–380, 2009.

[51] Dennis Abts, Michael R. Marty, Philip M. Wells, Peter Klausler, and Hong Liu. Energy proportional datacenter networks. In *Proceedings of the 37th annual international symposium on Computer architecture*, ISCA '10, pages 338–347, 2010.

[52] Priya Mahadevan, Sujata Banerjee, and Puneet Sharma. Energy proportionality of an enterprise network. In *Green Networking*, pages 53–60, 2010.

[53] Dushyanth Narayanan, Austin Donnelly, and Antony I. T. Rowstron. Write off-loading: Practical power management for enterprise storage. In *FAST*, pages 253–267, 2008.

[54] Wei Zheng, Ana P. Centeno, Frederic Chong, and Ricardo Bianchini. Logstore: toward energy-proportional storage servers. In *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, ISLPED '12, pages 273–278, 2012.

[55] Qingbo Zhu, Zhifeng Chen, Lin Tan, Yuanyuan Zhou, Kimberly Keeton, and John Wilkes. Hibernator: helping disk arrays sleep through the winter. In *Proceedings of the twentieth ACM symposium on Operating systems principles*, SOSP '05, pages 177–190, 2005.

[56] Eduardo Pinheiro and Ricardo Bianchini. Energy conservation techniques for disk array-based servers. In *Proceedings of the 18th annual international conference on Supercomputing*, ICS '04, pages 68–78, 2004.

[57] Seung Woo Son, Guangyu Chen, O. Ozturk, M. Kandemir, and A. Choudhary. Compiler-directed energy optimization for parallel disk based systems. *Parallel and Distributed Systems, IEEE Transactions on*, 18(9):1241–1257, 2007.

[58] David G. Andersen, Jason Franklin, Michael Kaminsky, Amar Phanishayee, Lawrence Tan, and Vijay Vasudevan. FAWN: A Fast Array of Wimpy Nodes. In *SOSP '09*, pages 1–14. ACM, 2009.

[59] Hrishikesh Amur, James Cipar, Varun Gupta, Gregory R. Ganger, Michael A. Kozuch, and Karsten Schwan. Robust and flexible power-proportional storage. In *Proceedings of the 1st ACM symposium on Cloud computing*, SoCC '10, 2010.

[60] HDFS Architecture. `http://hadoop.apache.org/common/docs/r0.20.0/hdfs_design.html`.

[61] Christopher Stewart and Kai Shen. Some joules are more precious than others: Managing renewable energy in the datacenter. In *Proceedings of the 2009 conference on Power aware computing and systems*, HotPower'09, 2009.

[62] Zhenhua Liu, Minghong Lin, Adam Wierman, Steven H. Low, and Lachlan L.H. Andrew. Geographical load balancing with renewables. *SIGMETRICS Perform. Eval. Rev.*, 39(3):62–66, December 2011.

[63] Zhenhua Liu, Minghong Lin, Adam Wierman, Steven H. Low, and Lachlan L.H. Andrew. Greening geographical load balancing. In *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, SIGMETRICS '11, pages 233–244, 2011.

[64] Krishna Kant. Distributed energy adaptive computing. In *ICC*, pages 1–5, 2010.

[65] Íñigo Goiri, Kien Le, Thu D. Nguyen, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. Greenhadoop: leveraging green energy in data-processing frameworks. In *Proceedings of the 7th ACM european conference on Computer Systems*, EuroSys '12, pages 57–70, New York, NY, USA, 2012. ACM.

[66] Íñigo Goiri, Ryan Beauchea, Kien Le, Thu D. Nguyen, Md. E. Haque, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. Greenslot: scheduling energy consumption in green datacenters. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 20:1–20:11, 2011.

[67] Kien Le, Ricardo Bianchini, Thu D. Nguyen, Ozlem Bilgir, and Margaret Martonosi. Capping the brown energy consumption of internet services at low cost. In *Green Computing Conference*, 2010.

[68] Baris Aksanli, Jagannathan Venkatesh, Liuyi Zhang, and Tajana Rosing. Utilizing green energy prediction to schedule mixed batch and service jobs in data centers. In *Proceedings of the 4th Workshop on Power-Aware Computing and Systems*, HotPower '11, pages 5:1–5:5, 2011.

[69] Steve Greenberg, Evan Mills, Bill Tschudi, Peter Rumsey, and Bruce Myatt. Best Practices for Data Centers: Lessons Learned from Benchmarking 22 Data Centers. 2006.

[70] Jichuan Chang, Justin Meza, Parthasarathy Ranganathan, Cullen Bash, and Amip Shah. Green server design: Beyond operational energy to sustainability. In *Proceedings of the 2010 international conference on Power aware computing and systems*, HotPower'10, 2010.

[71] Jason Flinn and M. Satyanarayanan. Managing battery lifetime with energy-aware adaptation. *ACM Trans. Comput. Syst.*, 22, May 2004.

[72] Intel Centrino Duo Processor Technology. *Intel Technology Journal*, 10(2), 2006.

[73] Bhavani Krishnan, Hrishikesh Amur, Ada Gavrilovska, and Karsten Schwan. VM power metering: feasibility and challenges. *SIGMETRICS Perform. Eval. Rev.*, 38:56–60, January 2011.

[74] D. Kusic, N. Kandasamy, and Guofei Jiang. Combined power and performance management of virtualized computing environments serving session-based workloads. *Network and Service Management, IEEE Transactions on*, 8(3):245 –258, September 2011.

[75] Antti P. Miettinen and Jukka K. Nurminen. Energy Efficiency of Mobile Clients in Cloud Computing. In *HotCloud'10*, 2010.

[76] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. Clonecloud: elastic execution between mobile device and cloud. In *Proceedings of the sixth conference on Computer systems*, EuroSys '11, pages 301–314, New York, NY, USA, 2011. ACM.

[77] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 49–62, 2010.

[78] Vinicius Petrucci, Orlando Loques, and Daniel Mossé. A framework for dynamic adaptation of power-aware server clusters. In *Proceedings of the 2009 ACM symposium on Applied Computing*, SAC '09, 2009.

[79] Mayank Raj, Krishna Kant, and Sajal K. Das. Energy adaptive mechanism for p2p file sharing protocols. In *Euro-Par Workshops*, pages 89–99, 2012.

[80] Ashish Sharma, Vishnu Navda, Ramachandran Ramjee, Venkata N. Padmanabhan, and Elizabeth M. Belding. Cool-tether: energy efficient on-the-fly wifi hot-spots using mobile phones. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, pages 109–120, New York, NY, USA, 2009. ACM.

[81] I. Kelenyi and J.K. Nurminen. Energy Aspects of Peer Cooperation Measurements with a Mobile DHT System. In *Communications Workshops, 2008. ICC Workshops '08. IEEE International Conference on*, pages 164 –168, may 2008.

[82] J.P. Sousa, V. Poladian, D. Garlan, B. Schmerl, and M. Shaw. Task-based adaptation for ubiquitous computing. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 36, May 2006.

[83] Takanori Watanabe. Acpi implementation on freebsd. In *Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference*, 2002.

[84] David S. Johnson, Alan J. Demers, Jeffrey D. Ullman, M. R. Garey, and Ronald L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput.*, 3(4):299–325, 1974.

[85] Chandra Chekuri and Sanjeev Khanna. On multi-dimensional packing problems. In *SODA*, pages 185–194, 1999.

[86] D K Friesen and M A Langston. Variable sized bin packing. *SIAM J. Comput.*, 15(1):222–230, 1986.

[87] Krishna Kant. *Introduction to computer system performance evaluation.* McGraw-Hill, 1992.

[88] VMWare ESX Server. `http://www.vmware.com/products/vsphere/esxi-and-esx/index.html`.

[89] Extech Power Analyzer. `http://www.extech.com/instruments/categories.asp?catid=14`.

[90] VMWare Dynamic Power Management. `http://www.vmware.com/virtualization/green-it`.

[91] VMWare vSphere. `http://www.vmware.com/products/storage-vmotion/overview.html`.

[92] M. Arlitt and T. Jin. 1998 World Cup Web Site Access Logs. `http://www.acm.org/sigcomm/ITA/`.

[93] Luiz André Barroso, Jeffrey Dean, and Urs Hölzle. Web search for a planet: The google cluster architecture. *IEEE Micro*, 23, March 2003.

[94] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, 2008.

[95] Bhuvan Urgaonkar, Giovanni Pacifici, Prashant Shenoy, Mike Spreitzer, and Asser Tantawi. An analytical model for multi-tier internet services and its applications. In *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 2005.

[96] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 3rd edition, 2009.

[97] Edmund B. Nightingale, Jeremy Elson, Jinliang Fan, Owen Hofmann, Jon Howell, and Yutaka Suzue. Flat datacenter storage. In *Proceedings of the 10th*

USENIX conference on Operating Systems Design and Implementation, OSDI'12, pages 1–15, 2012.

[98] Cezary Dubnicki, Leszek Gryz, Lukasz Heldt, Michal Kaczmarczyk, Wojciech Kilian, Przemyslaw Strzelczak, Jerzy Szczepkowski, Cristian Ungureanu, and Michal Welnicki. Hydrastor: a scalable secondary storage. In *Proccedings of the 7th conference on File and storage technologies*, FAST '09, 2009.

[99] Austin T. Clements, Irfan Ahmad, Murali Vilayannur, and Jinyuan Li. Decentralized deduplication in san cluster file systems. In *Proceedings of the 2009 conference on USENIX Annual technical conference (ATC)*, 2009.

[100] Keren Jin and Ethan L. Miller. The effectiveness of deduplication on virtual machine disk images. In *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference*, 2009.

[101] Chun-Ho Ng, Mingcao Ma, Tsz-Yeung Wong, Patrick P. C. Lee, and John C. S. Lui. Live deduplication storage of virtual machine images in an open-source cloud. In *Proceedings of the 12th ACM/IFIP/USENIX international conference on Middleware*, 2011.

[102] Benjamin Zhu, Kai Li, and Hugo Patterson. Avoiding the disk bottleneck in the data domain deduplication file system. In *FAST '08: Proceedings of the 6th USENIX Conference on File And Storage Technologies*, 2008.

[103] Zhichao Li, Kevin M. Greenan, Andrew W. Leung, and Erez Zadok. Power consumption in enterprise-scale backup storage systems. In *Proceedings of the 10th USENIX conference on File and Storage Technologies*, FAST'12, 2012.

[104] S. Quinlan and S. Dorward. Venti: a new approach to archival storage. In *First USENIX conference on File and Storage Technologies*, Monterey,CA, January 2002.

[105] Lawrence You, Kristal Pollack, and Darrell D. E. Long. Deep store: An archival storage system architecture. In *ICDE'05: Proceedings of the 21st International Conference on Data Engineering*, April 2005.

[106] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, 2003.

[107] Carl A. Waldspurger. Memory resource management in vmware esx server. In *OSDI*, 2002.

[108] Partho Nath, Michael A. Kozuch, David R. O'Hallaron, Jan Harkes, M. Satyanarayanan, Niraj Tolia, and Matt Toups. Design tradeoffs in applying content addressable storage to enterprise-scale systems based on virtual machines. In *ATEC '06: Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, 2006.

[109] Ahmed El-Shimi, Ran Kalach, Ankit Kumar, Adi Oltean, Jin Li, and Sudipta Sengupta. Primary data deduplication-large scale study and system design. In *Proceedings of the 2012 USENIX conference on Annual Technical Conference (ATC)*, 2012.

[110] Jacob Gorm Hansen and Eric Jul. Lithium: virtual machine storage for the cloud. In *Proceedings of the 1st ACM symposium on Cloud computing (SoCC)*, 2010.

[111] FUSE - Filesystem in User Space. `http://fuse.sourceforge.net/`.

[112] Apache Cassandra. `http://cassandra.apache.org/`.

[113] Kiran Srinivasan, Tim Bisson, Garth Goodson, and Kaladhar Voruganti. idedup: latency-aware, inline data deduplication for primary storage. In *Proceedings of the 10th USENIX conference on File and Storage Technologies*, FAST'12, 2012.

[114] Flexible IO Tester. `http://git.kernel.dk/?p=fio.git;a=summary`.

[115] National Renewable Energy Laboratory (NREL). `http://www.nrel.gov/`.

[116] A.-H. Mohsenian-Rad and A. Leon-Garcia. Coordination of cloud computing and smart power grids. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 368–372, 2010.

[117] Mahdi Ghamkhari and Amir Hamed Mohsenian Rad. Optimal integration of renewable energy resources in data centers with behind-the-meter renewable generator. In *ICC*, pages 3340–3344, 2012.