HOLISTIC POWER OPTIMIZATION FOR DATACENTERS

A Dissertation
Presented to
The Academic Faculty

By

Sungkap Yeo

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in
Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
May 2015

HOLISTIC POWER OPTIMIZATION FOR DATACENTERS

Approved by:

Dr. Thomas Conte, Advisor
Professor,
School of ECE and Computer Science
Georgia Institute of Technology

Dr. George F. Riley
Professor,
School of ECE
Georgia Institute of Technology

Dr. Sudhakar Yalamanchili
Professor,
School of ECE
Georgia Institute of Technology

Dr. Hyesoon Kim
Associate Professor,
School of Computer Science
Georgia Institute of Technology

Dr. Karsten Schwan
Professor,
School of Computer Science
Georgia Institute of Technology

Date Approved: Dec 2014

To my family.

ACKNOWLEDGMENT

First of all, I would like to thank my research advisor Dr. Thomas Conte for his continuous and priceless support. I would also like to thank Dr. Sudhakar Yalamanchili, Dr. Karsten Schwan, Dr. George F. Riley, and Dr. Hyesoon Kim for serving as my dissertation committee members.

I am also thankful to my former research advisor Dr. Hsien-Hsin S. Lee and all Georgia Tech colleagues, Jaewoong Sim, Dr. Nak Hee Seong, Hanseung Lee, Jen-Cheng Huang, Mohammad Moazzem Hossain, Lifeng Nai, Tzu-Wei Lin, Dr. You-Chi Cheng, Dr. Chinnakrishnan Ballapuram, Dr. Mrinmoy Ghosh, Dr. Richard Yoo, Dr. Dong Hyuk Woo, Dr. Dean Lewis, Eric Fontaine, Manoj Athreya, Andrei Bersatti, Hyojong Kim, Dr. Sunpyo Hong, Dr. Minjang Kim, Dr. Jaekyu Lee, Dr. Hyungwook Kim, Dr. Jaegul Choo, and Dr. Ilseo Kim. A special thanks goes to Hanseung Lee for helping me in validating analytical models, Jen-Cheng Huang for continuously encouraging me in developing research ideas, Dr. Nak Hee Seong for helping me in my recent non-volatile memory studies, and Jaewoong Sim for continuously motivating and challenging me with insightful feedback.

I have been fortunate enough to meet and co-work with professionals in industry as well. Hoeju Chung of Samsung Electronics helped me better understand internal circuit design of phase change memory. Endless rounds of discussions with Dr. Taemin Kim of Intel Corporation and Dr. Kisun You of Apple Incorporated motivated me to study microarchitecture designs.

I participated in one of the most exciting projects, Green Electricity Network Integration, which was supported by ARPA-E, in my last year at Georgia Tech. I would like to thank Dr. Santiago Grijalva, Dr. Masoud H. Nazari, Mitch Costley, M. Javad Feizollahi, Jennifer Howard, and Umer Tariq for successfully leading the project.

I am thankful to my parents, Youngjin Lee and Hoyoung Yeo, and my sister, Dongeun

Yeo, who have always been my number one fan. Their support was one of the most powerful motivations for me. Lastly, I must thank my wife, Yejin Kim, who has long been the pillar of hope for me. Without her endless support and unconditional love, I would not be able to return to the school.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The purpose of this dissertation describes several power optimization techniques for energy efficient datacenters. To achieve this goal, it approaches power dissipation holistically for entire datacenters and analyzes them layer-by-layer from (1) the infrastructure level, (2) the system level, and all the way down to (3) the micro-architecture level.

First, for infrastructure-level power optimization of datacenters, this work presents infrastructure-level mathematical models. These models demonstrate that to achieve optimal performance in a heterogeneous cloud infrastructure, the response time of the slowest node should be no more than three times as long as that of the fastest node. This dissertation also presents a holistic warehouse-scale datacenter power and performance simulator, SimWare. To optimize datacenter energy efficiency, SimWare analyzes the power consumption of servers, cooling units, and fans as well as the effects of heat recirculation and air supply timing. Experiments using SimWare show a high loss of cooling efficiency resulting from the non-uniform inlet air temperature distribution across servers.

Second, this study describes a system-level technique, ATAC, for power efficient datacenters. The SimWare framework reveals that only a small number of servers at hot spots suffer from high inlet air temperature, and cooling these servers largely compromises cooling efficiency. Thus, to tackle these inefficiencies, this dissertation proposes ambient temperature-aware capping, ATAC, which maximizes power efficiency while minimizing overheating.

Finally, this dissertation describes a micro-architecture level technique under the context of emerging non-volatile memory technologies. Non-volatile solid-state memory technologies often exploit the analogous characteristics of an underlying material that stores more than one bit per cell. We first show that storing more than one bit per cell, or multiple bits per cell, ends up with much higher soft-error rates than conventional technologies. However, multi-bit per cell technology can still be used as approximate storage. To this

end, we propose a new class of multi-bit per cell memory in which both a precise bit and an approximate bit are located in a physical cell.

With the development of these techniques, the contribution of this body of work is a reduction in the power consumption of datacenters in a holistic way, eliminating one of the most important hurdles to the proliferation of cloud-computing environments.

CHAPTER 1

INTRODUCTION

The current de-facto future computing model for all types of computing is the concept of cloud computing. Ideally, moving computing to the cloud relieves much of the responsibility of users by providing higher reliability and availability for data computation and management. With this transformational paradigm shift, the main computing power and resources will be provided by cloud service providers that maintain and operate a complete infrastructure, solution platforms, and a plethora of applications in the so-called *datacenters*. Datacenters accommodate computing nodes and peripherals that consume electrical power for computing and cooling facilities in units of megawatts. For example, in 2010, the world's largest online game, World of Warcraft, developed by Blizzard Entertainment, required more than 20,000 systems with more than 75,000 processing cores for their online services. Aside from the cost of building the infrastructure of a datacenter, energy costs for operating and cooling these power-hungry datacenters have reached a level that surpasses hardware acquisition costs. In 2011, datacenters, accounting for $27 billion in annual electricity cost [1, 2], consumed a total of 1.5% of energy worldwide. With the rapid growth of cloud-based services, the upward trend is expected to continue with energy consumption by datacenters estimated to double by 2014.

As the cloud computing model becomes more pervasive, the power consumption of datacenters will continue to increase as the number of online users rises worldwide. Such increased power usage is not simply an economic concern for service providers, datacenter operators, and end users; it is also environmental concern, for generating this large amount of energy also inevitably leads to more carbon dioxide emissions, which accelerate pollution and global warming. Therefore, operating datacenters at maximal power efficiency has become a top priority of scientists, engineers, and policy makers in myriad multi-disciplinary areas. However, before any effort is devoted to this issue, researchers

and policy makers need to fully understand the entire power delivery and distribution system; that is, they must be able to answer the following question: Where does the power consumed by datacenters go?

This dissertation takes a holistic view of power dissipation for the entire datacenter and analyzes them layer-by-layer on the infrastructure level, the system level, and all the way down to the micro-architecture level. It begins by discussing the power breakdown of each level using data available in the public domain and then proposes innovative techniques for each level.

In general, infrastructure-level electrical power usage falls into two categories: computing and cooling. Legacy datacenters often consume more than 50% of their total power for cooling [3] while state-of-the-art datacenters consume less than 10% [4]. A metric referred to as power usage effectiveness, or PUE [5], was proposed to measure the efficiency of the datacenter infrastructure. However, this metric could be misleading because PUE ignores the increased fan power that occupies the non-negligible portion of power consumption by servers [6]. When an administrator decides to reduce the power consumption of air conditioning (CRAC) units in the computing room, the fans in servers will blow harder and consume more power than before, resulting from higher room temperature than that during normal operation. In other words, raising the room temperature of a datacenter always results in lower PUE than before because of both decreased cooling power and increased server power. Although increased server power comprises a large portion of total datacenter power, it has not been accounted for in the PUE metric.

Prior studies have proposed software tools that simulate datacenters; however, they were not complete because the tools were lacking critical parameters. For example, CloudSim [7] and DCSim [8] did not include the effect of increased fan power and heat recirculation. Other studies [9, 10, 11, 12] largely ignored the air-travel time from CRAC units to servers. To address these shortcomings, this dissertation introduces a new datacenter simulator, SimWare, with detailed temperature, power, and performance models for servers and CRAC

units. It also simulates the heat-recirculation effect and the detailed timing model for the travel time of supply air.

This dissertation also proposes a system-level technique that saves a significant amount of the cooling power of datacenters with negligible performance overhead. The aforementioned holistic datacenter simulator reveals that not all server locations in a datacenter are identical in terms of cooling: Some suffer from high temperatures while the others are not. More specifically, server locations at the highest position in racks are identified as hot spots, and about 70% of cooling power is used for cooling down these servers at hot spots. If a system-level technique prevents CPUs from temperature emergencies, datacenters can save a significant amount of cooling power. Motivated from these observations, this dissertation proposes a new thermal optimization technique that only triggers performance capping for servers at hot spots. In other words, the new technique is designed to exploit the inequality, or non-uniformity, of the inlet-air temperature among the servers in a rack.

The last contribution of this dissertation is the proposal of a micro-architectural technique for power-efficient datacenters. Datacenters today run a variety of workloads including error-tolerant approximate workloads such as voice recognition or image processing. Approximate computing is a promising way to provide energy efficiency for such types of applications that require precision. As approximate computing embraces imprecision, however, it is crucial for streamlining computational resilience against errors for the best tradeoff among accuracy, performance, and energy consumption. Therefore, this dissertation discusses error resiliency in the context of approximate solid-state memory. More specifically, it provides a comprehensive study to efficiently enable phase-change memory (PCM) as approximate storage. It is shown that simply relaxing a write-and-verify sequence in cell programming does not provide good error resilience. Therefore, this dissertation proposes a new class of multi-level PCM cells for approximate storage, in which a precise bit and an approximate bit are co-located (*i.e.,* half-precise/half-approximate) in a PCM cell.

The rest of this document is organized as follows. The following chapter discusses the origin and history of the problem as well as state-of-the-art techniques in different levels, infrastructure-level, system-level, and micro-architecture-level techniques. The next three chapters, Chapter 3 through Chapter 5, present novel optimization techniques for these levels. More specifically, Chapter 3 discusses the infrastructure-level power breakdown of datacenters and presents analytical models that can be used to optimize the energy efficiency of naturally heterogeneous datacenters. In addition, this chapter also presents a holistic datacenter simulator that takes the critical power-consuming components of datacenters into account. Chapter 4 discusses the system-level power breakdown of a server and presents a system-level power optimization technique, ATAC. Chapter 5 also shows the micro-architecture-level power breakdown of a CPU first and then proposes a class of multi-level PCM cells for power efficient and reliable computing. Lastly, Chapter 6 concludes the dissertation.

CHAPTER 2

ORIGIN AND HISTORY OF THE PROBLEM

Power optimization is one of the most active research areas in several engineering disciplines for the last decade. Moore's Law continues to drive a large number of transistors to be integrated on a single chip, and these transistors consume exponentially increased dynamic power. On the other hand, device miniaturization increases the operating frequency at the expense of increased dynamic power and, at the same time, worsens the leakage power. Technologies at the device level (e.g., Intel's high-k metal gate in their 45nm process) all the way up to the design of a datacenter all aim at minimizing power consumption. For example, datacenters save millions of dollars paid for energy even with a small percentage of improvement in reducing power consumption. This section discusses origin and history of power optimization problems from a hierarchical perspective starting from the infrastructure, system, and finally the micro-architecture level.

2.1 Infrastructure-level Techniques

2.1.1 Energy-Proportional Computing

In typical datacenters, the average utilization is known to be as low as 20% to 30% [13]. One reason for this low utilization is that since datacenters are prepared to serve the highest demand of a day or a week, their computing power is over-provisioned to satisfy the worst-case scenario even when the average number of requests is low. Given the low utilization of a datacenter by its nature, the need for energy-proportional computing [14] has risen. The basic concept of the energy-proportional computing is that when the utilization of a computing node is under 100%, say 50%, the power consumption of the computing node should be half the power of 100% utilization. To apply this concept to a datacenter, an energy-proportional datacenter with 30% utilization should consume only 30% of its peak power. However, the energy-proportional computing concept is not ready to the vast majority of today's equipments. A power model for today's common computing node shows

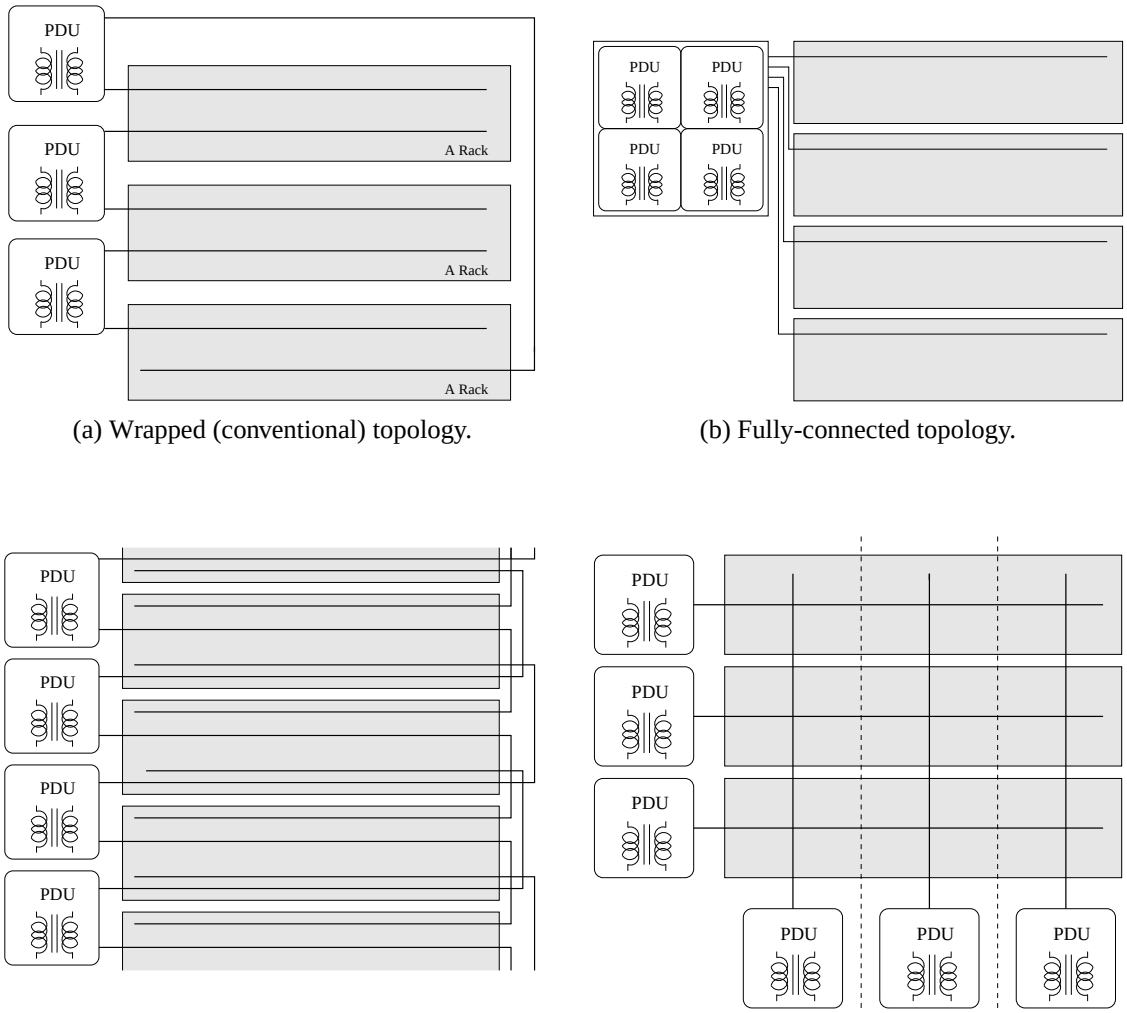
that the computing node consumes almost half of its peak power when it is completely idle (0% utilization) and consumes about 75% of its peak power when utilization is 50% [14].

To alleviate this problem, a new idea has been proposed for datacenters with common equipments [15]. In this work, by considering that even common equipments have a nearly energy-proportional characteristic at high utilization, some computing nodes are suggested to be turned off to keep the others busy. For example, when ten computing nodes of the same type are around 5% utilization, the idea suggests to turn nine machines off but keeping only one node up and running. In the ideal situation of this technique, the aggregate power consumption can be meaningfully close to the utilization even with non-energy-proportional machines.

2.1.2 Power Routing

Power Routing [16] is a technique for reducing redundant power delivery infrastructure. In high-availability datacenters, more than one power distribution units (PDU) are used for supporting a server cluster to reduce the risk of PDU failure. In the event of PDU failure, other PDUs take over the duty of the broken PDU to support uninterrupted service. Hence, high availability and reliability in datacenters can be achieved via such over-provisioning to provide reserved capacity. The amount of the reserved capacity that causes overhead in power delivery infrastructure highly depends on the topology used by the datacenter. For example, in the wrapped topology illustrated in Figure 1a, two PDUs can be brought in to recover a single PDU failure. In other words, each PDU in the wrapped topology needs to have 50% of the reserved capacity for recovering a single PDU failure. On the other hand, when it comes to a single PDU failure, an example of a fully-connected topology as shown in Figure 1b can be used to have three additional PDUs for replacing one failed PDU. In this case, the amount of redundant capacity that each PDU must have is 33% of the peak power a rack can draw.

The design rationale of Power Routing is that depending on the connectivity among PDUs and server clusters in a datacenter, the reserved capacity can vary for recovering a



(a) Wrapped (conventional) topology.

(b) Fully-connected topology.

(c) Serpentine topology.

(d) X-Y topology.

Figure 1: Power distribution topologies for Power Routing.

PDU failure. Because reserved or redundant capacity in PDUs directly indicates that more money is to be spent on power-delivery infrastructure than the PDUs without redundancy, it is important to choose a routing topology without redundancy while maintaining the same level of scheduling ability. Power Routing is one of such techniques. Power Routing comprises two parts. First, this idea introduced many different topologies between PDUs and server clusters such as the serpentine topology in Figure 1c or the X-Y topology in Figure 1d. Second, Power Routing introduced a heuristic scheduling algorithm for assigning a

power line to servers while balancing loads. As this power assignment is a non-polynomial (NP) problem, authors first let the servers be fractionally assigned to the power feeds by using standard linear programming methods. From this fractional solution, the real problem will be solved approximately. When the approximate solution fails to meet the requirements from PDU specs or fails to balance between AC phases, they repeat the second step. By applying real datacenter power traces to this idea, Power Routing could save 5% to 10% of the required power capacity for conventional datacenters and 22% to 28% for the energy-proportional servers.

2.2 System-level Techniques

2.2.1 DRAM Power Management

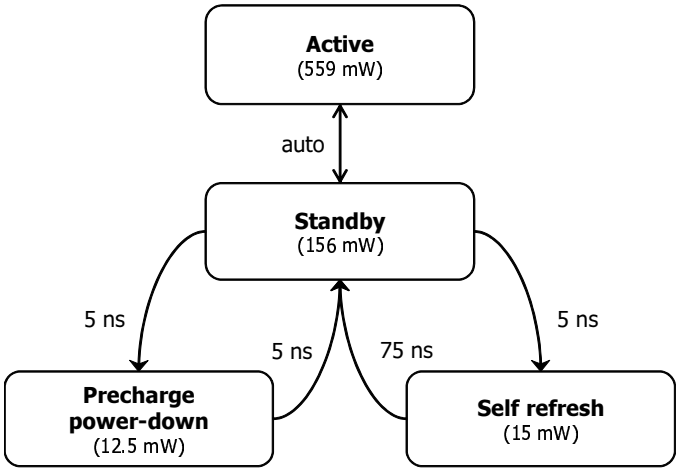


Figure 2: Operating modes for DDR DRAM [17]

The main memory made of dynamic random access memories (DRAM) is a power hog as demonstrated in Figure 21. To save DRAM power, modern DRAM supports up to six different power states for RDRAM [18] or four different power states for double data rate (DDR) DRAM [17]. More specifically, a DRAM controller can put an entire rank [1] of the main memory into the low power state if the rank has not been used for a given period of

[1] In DRAM, a rank is uniquely addressable 64 bits or 72 bits (when supporting 8 bits error correction code) data area. In a dual rank memory module, for example, memory controller uses chip select signal to choose what rank to access. In other words, the memory controller can access only half of the entire memory space in a cycle.

time. However, when a rank is in the low power state, there will be non-negligible delay before it becomes ready to be read or written again. Figure 2 illustrates this cycle. There are four power modes implemented in current DDR DRAM [17]. When a rank is in the standby mode, it is automatically moved to the active mode when a read or write request arrives. On the contrary, a transition to the other two modes, self-refresh or power-down mode, is done manually by the memory controller. The power-down mode starts when the memory controller lowers the clock enable signal (CKE) to the idle DDR DRAM rank, and the self-refresh mode starts when CKE is lowered as well as the auto-refresh signal is sent. These two low power modes are essentially similar in terms of power savings, however, different in terms of allowed interval in each mode. For the power-down mode, a rank can not be in this mode more than maximum refresh interval, because no refresh signal is sent to a rank in this mode. In contrast, a rank can be in the self-refresh mode without time limit, because the on-chip timer in DRAM generates periodic refresh signal for a rank in this mode. This is why the self-refresh mode has longer transition delay and requires slightly more power than the power-down mode. To make use of these different power states for saving power in DRAM, Hur *et al.* [19] proposed a simple power-down policy. First, each rank of the main memory has a counter that resets upon every read or write request and increases upon every idle cycle for bookkeeping the number of idle cycles for the rank. Second, when the counter reaches a threshold value, the memory controller checks the internal queue to verify whether there is a read or write request for this rank. If a rank has been idle for more than the threshold time and there is no read or write request in the queue, the memory controller puts the rank into the power-down mode. This policy is reported to increase DRAM energy efficiency by 11% to 43% for different benchmark programs.

2.2.2 Powernap

On the other hand, Powernap [20] has been proposed for eliminating the idle power of servers. The basic idea of Powernap starts from the fact that once a server becomes idle, the

average idle time is around $100ms$ for most of internet services while some other services (domain-name services or scientific computing clusters) have longer average idle time than the others up to one or several seconds. For these reasons, if a server can be turned off and brought back in a few milliseconds, the server can effectively be turned off during its idle period. For this fast transition between full performance and *nap* modes, Powernap suggests to use the S3 sleep state (also known as standby state) for CPUs, the self-refresh technique for DRAM, solid state disks (SSD) for storage devices, and the wake-on-LAN technique for network interface cards. By using these features, a typical blade can change its power state from full performance mode to the *nap* mode in $300\mu s$ and vice versa. With the penalty of less than $1ms$ transition time, a typical server that consumes 270W when idle and 450W when active can save significant power while in the *nap* mode because it consumes only 10W during the nap mode. Further comparison between Powernap and dynamic voltage-frequency scaling (DVFS) technique showed that Powernap technique with less than $10ms$ of transition time always outperforms DVFS in terms of response time and power scaling. As a result, Powernap yields a steep power reduction up to 70% for internet servers.

2.2.3 Power Capping

Power Capping [21] is another system-level technique that guarantees the power consumed by a server to be confined within a given power envelope, or the capped value. For example, if a server with power capping capability is set to $200W$, the power controller inside the server will keep the power consumption of this server below $200W$. To achieve this design goal, the controller throttles performance by using DVFS technique when it consumes more power than the capping value. The closed-loop feedback controller for Power Capping is illustrated in Figure 3. First, the controller is set to a certain value representing the maximum allowed power budget for this server. The controller calculates the ideal throttle level based on the set point and the measured power consumption. Second, the actuator, a first-order delta-sigma modulator, calculates the target throttle level based on the ideal

and real throttle level retrieved from other sources. By using this extra controller on top of the conventional power supply design, a server can safely be under-provisioned, the key to enhance efficiency of the power delivery infrastructure.

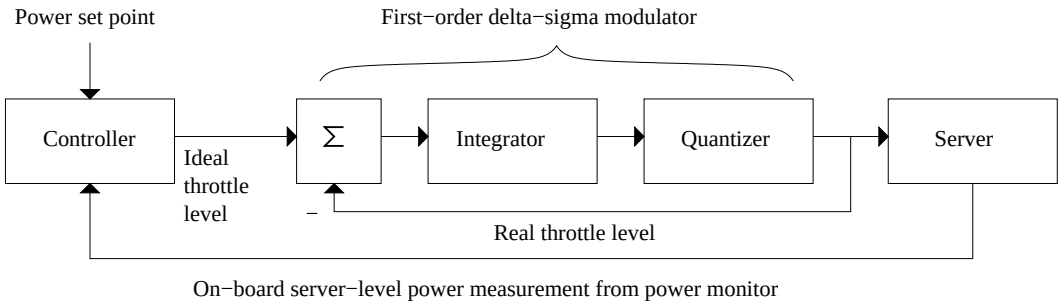


Figure 3: System diagram for Power Capping controller.

2.2.4 Dynamic Voltage-Frequency Scaling (DVFS)

Dynamic voltage-frequency scaling is a technique for reducing the dynamic active power by lowering the operating voltage and/or frequency of a microprocessor. The active power of a CMOS circuit is linearly and quadratically proportional to the frequency (f) and the operating voltage (V_{dd}), respectively. In other words,

$$\text{Active Power} \propto V_{dd}^2 \cdot f. \tag{1}$$

Therefore, for certain instances such as when the utilization of a processor is low, when the response time is insensitive, or when the running tasks are not critical, a system with the DVFS technique can reduce its operating voltage and frequency on the fly with minimal impact to the quality of service. Although the voltage and frequency can be controlled independently in a typical microprocessor, it is common to use a low voltage for a low frequency. This is because when using a low operating voltage, the time for charging any given capacitor takes longer than the baseline with a high operating voltage. As a result, a low voltage leads to a slower operation or slower operating frequency than the baseline. In all, the main drawback of this technique is that a low voltage and frequency can inadvertently penalize the performance.

2.2.5 Clock Gating and Power Gating

Distributing the clock signal across the entire die area in synchronous circuits requires more than one third of the total chip power. It gets worse if a chip uses a metal grid clock distribution network for minimizing the clock skew as discussed earlier. For reducing the active power for the clock distribution network, the most commonly used technique is clock gating. The basic idea of clock gating is to cut off the clock signal for the regions that are not used. When the clock signal does not enter a particular region of a circuit, it avoids the switching activities of its flip-flops and clock buffer tree, thereby saving power. To achieve this goal, two types of solutions are employed: a latch-free clock gating and a latch-based clock gating. In the latch-free clock gating design, a simple two-input AND gate is used to enable or disable the clock signal while the latch-based design uses a level-sensitive latch for holding the enable signal. Whenever the enable signal is off, the delivery of the clock signal is cut off. The main drawback of this clock gating is that the additional combinational logic will likely elongate the propagation delay in delivering clock signal to all corners of a chip. Due to this extra propagation time that exacerbates the clock skew, a circuit with clock gating may reduce the operating frequency.

Although clock gating can help reduce the active power of unexercised circuits, this cannot save leakage power. As the leakage power continues to worsen when the feature sizes shrink due to lowered threshold voltage (as shown in Figure 35), power gating is introduced to disconnect the unused circuits from the power source using a sleep transistor with a high threshold voltage to eliminate the leakage current. Figure 4a illustrates an example of a sleep transistor that gates off the power supply path via V_{ss} of an SRAM cell. This more aggressive power-saving technique faces several drawbacks if not used wisely. First, power-gating a circuitry, from active to inactive or vice versa, takes time in order to stabilize the circuit operation. Depending on the scale of the circuit block, the circuit may need to be switched off in multiple steps to keep the ground bounce noise under safety margin. Hence, it could affect the overall performance. Second, switching the states

consumes extra power. For these reasons, when and where to power off must be chosen carefully. In other words, power gating should be performed only when the penalty in power and time for turning on and off is significantly less than the power that can be saved.

2.3 Micro-architecture-level Techniques

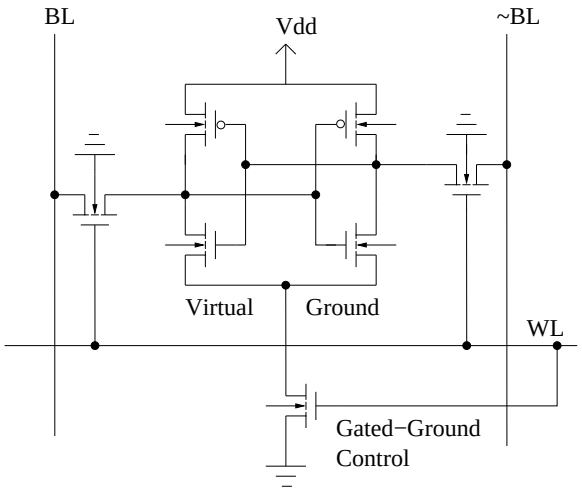
Microarchitectural power reduction techniques have been an active research area among processor architects. A majority of these studies focus on on-chip memories, *i.e.*, caches. Some techniques combine circuit and microarchitectural optimization techniques to reduce power. Subsequent sections review some major tasks toward these efforts.

2.3.1 Reconfigurable Caches

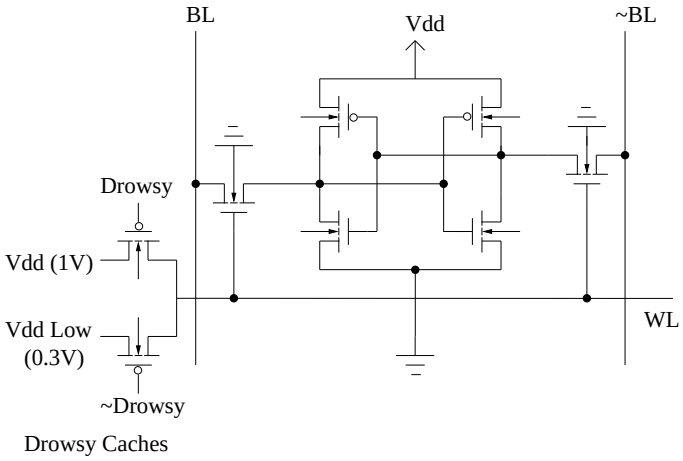
Selective cache ways is one of the earliest architectural techniques proposed for reducing power consumption in caches of a processor. It selectively turns off a subset of cache ways for an associative cache at run-time. The idea starts from the fact that large on-chip caches are usually partitioned into several subarrays for reducing latencies. Because each subarray effectively stores one data cache way, it can readily be turned off at the hardware level. The mechanism can be supported with minimal additional hardware— a Cache Way Select Register (CWSR), to store which cache ways to use, and special instructions for reading and writing the CWSR. An application can disable selected cache ways during the period of modest cache activities without much performance impact. As shown in [22], this on-demand cache resource allocation mechanism saves 40% in overall cache energy dissipation in a four-way set associative cache with less than average 2% performance penalty.

2.3.2 Cache Decay

Given the trend of integrating larger and larger on-die caches continues, researchers have studied and proposed various techniques to control the leakage power of these components. *Cache decay* [23] is one of such techniques that combine power-supply-gating shown in Figure 4a with dynamic architectural behavior for controlling the leakage power



(a) Power gating in Cache Decay



(b) Drowsy Caches

Figure 4: Circuit diagram for Cache decay and Drowsy cache

of each individual cache line. It was motivated by the observation that cache lines are "dead" for more than 70% of the time. The dead time of a cache line is defined as the time of its last access and the time it is evicted. To avoid leakage power consumed during the dead time, if one can predict a cache line is dead, the line can be evicted and powered off earlier than the actual replacement taking place. The prediction is achieved by employing a decay counter for each cache line to book-keep the idleness of the line. When the down-counter reaches zero indicating the line is not being accessed for a given threshold, the line will be early-evicted and enter the power-off state using power-gating to save leakage

energy.

One drawback of the cache decay technique is the potential performance loss due to the fact that early power gating loses cache data, which may causes additional cache misses. Therefore, "when to decay a cache line" becomes critical. This work experimented different decay intervals from 1k cycles to 512k cycles and showed that a decay interval of 8k cycles showed the best saving result with a 70% reduction of the leakage power.

2.3.3 Drowsy Caches

Drowsy cache [24] was proposed to ameliorate the performance issue due to data loss of cache decay. In a drowsy cache, a cache line can choose between two different supply voltages, a normal voltage (V_{dd}) for regular cache lines, and a lowered one (V_{dd-low}) for drowsy cache lines. When a line is put into the drowsy mode, the data content is preserved although it has to pay a slight penalty (one to two cycles) to reinstate the line back to normal operated voltage before it can be re-accessed. Cache lines with the scaled down supply voltage can significantly reduce the leakage current by 6x to 10x due to short-channel effects. For the drowsy cache technique, there are additional hardware overheads. First, a drowsy bit is added to each cache line to indicate whether the cache line is in drowsy mode or not. Second, a voltage controller is added as illustrated in Figure 4b to supply a normal voltage for active state cache lines and a lowered voltage for drowsy state cache lines. Third, the word line gating circuit is added to prevent direct access to drowsy cache lines. With these additional hardware overheads, cache lines periodically change its state to the lower power one, and the line is woken up in the penalty of one cycle when it has to be accessed. Due to the overhead of additional cycle to wake up a cache line, performance could be degraded as much as 2% with an average of less than 1%. With this small impact on performance, the total energy (including static and dynamic) consumed in cache lines were reduced by 75%.

2.3.4 Razor

Razor [25], a combination of micro-architectural and circuit-level techniques, can substantially reduce the power consumption of a microprocessor by aggressively adopting sub-critical voltage in the pipeline. Similar to DVFS, Razor dynamically lowers the operating voltage to significantly reduce power consumption. However, even with the DVFS technique, there are voltage margins to be obeyed to avoid any execution error in the processor. For example, there have to be a process margin to consider manufacturing variations, an ambient margin to prevent processors from malfunctioning due to high temperature, and a noise margin to tolerate various unknown noise sources. Without these voltage margins, a processor could generate incorrect computation results mostly due to timing failure in the slow latches. In many cases, these margins are over-estimated to guarantee a reasonably large guard band for correctness. The design rationale of Razor challenged this worst-case design constraint and proposed to aggressively and dynamically scale down the operating voltage until an error is detected. Once an error is detected, a recovering mechanism will be triggered to correct these errors dynamically. As such, Razor can approach the minimal power consumption by lowering the supply voltage to the lowest possible value. The error detection mechanism is achieved by employing a *shadow latch* with a delayed clock to each normal flip-flop. As shown in Figure 5, the shadow latch with the delayed clock is designed to ensure to latch the correct incoming data while the normal flip-flop could fail due to too aggressive dynamic voltage scaling (DVS). Whenever the value of the shadow latch mismatches the value in the DVS-ed flip-flop, a timing error is indicated. Then, a pipeline flush and replay similar to branch mis-prediction recovery will follow with incrementally increased supply voltage. This supply voltage feedback control system will eventually reach the optimal operating voltage for a specific processor that runs a specific application. As shown in the original study [25], the error detecting circuits with aggressive DVS can reduce the power consumption of a processor by 64% with 3% performance impact [26].

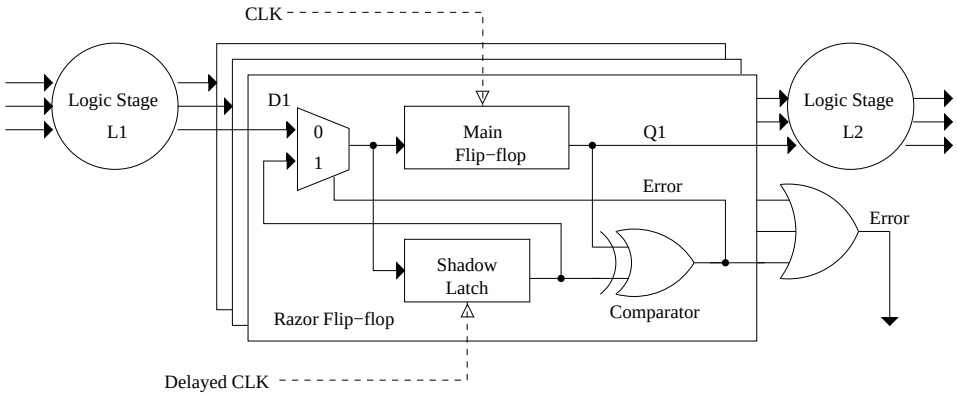


Figure 5: Razor flip-flop employs a shadow latch with delayed clock.

2.4 Challenges in Power Optimization

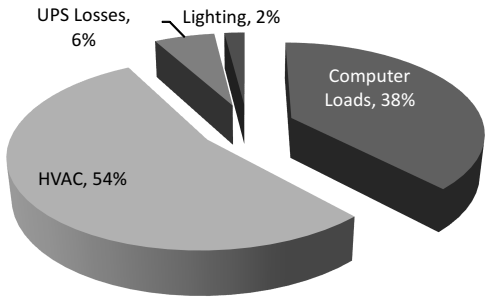
Although the power consumption of targeted components can be improved with the power optimization techniques discussed above, it does not guarantee an overall saving when they are applied altogether. Under certain circumstances, the savings of individual optimization techniques are not additive, worse yet, they could cancel each other out. Therefore, whenever a new power optimization is being considered, it must be thoroughly evaluated together with all existing solutions applied to the datacenter.

One common pitfall in power optimization is so-called *balloon effect*. In the balloon effect, suppressing one corner of a balloon may inadvertently inflate the other side. Similarly, saving power on one particular component may increase power consumption of others in the system. For example, administrators could raise the room temperature of datacenters for saving cooling energy; however, such optimization may result in increased fan power in the servers as raising the room temperature makes inlet-air temperature of servers higher than before. Without proper trade-off evaluation prior to the optimization, the overall facility power may end up being increased rather than reduced. Therefore, the proposed research introduces a holistic datacenter simulator on the following section.

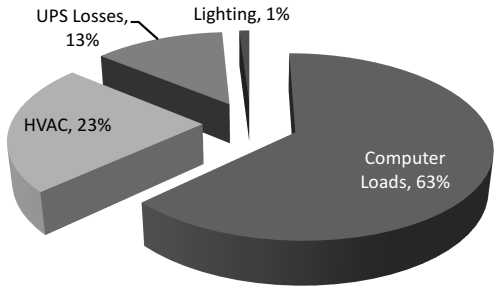
CHAPTER 3

INFRASTRUCTURE-LEVEL OPTIMIZATION

3.1 Infrastructure-level Power Breakdown



(a) datacenter 8.1 (total 578kW)



(b) datacenter 8.2 (total 1681kW)

Figure 6: Power breakdown of two different datacenters [3]

This chapter first analyzes the power distribution from the perspective of the highest level, *i.e.*, the infrastructure level, by taking published data from actual datacenters. When the electrical power from a power plant is delivered to a datacenter, it is consumed to operate two main facilities. Firstly, it powers up all the computing equipments and hosts the computing services. Secondly, as these computing devices convert the supplied power into useful computation and dissipate heat, the datacenter has to arrange additional power to remove the heat generated from the facility. These computing nodes and their cooling

system are two major power consumers in a typical data center. In addition, a datacenter also uses power for their power delivery infrastructures including the uninterrupted power supply units (UPS) or other supplementary infrastructures, *e.g.*, lighting. In addition to the utility power, modern datacenters typically build their own power generators along with UPS systems in order to guarantee a stable, uninterrupted power supply system.

The breakdown of power usage of these components at the infrastructure level is illustrated in Figure 6. The data were taken from a case study performed by the Lawrence Berkeley National Lab [3]. In Figure 6, the portion, "Computer Loads," accounts for the power drawn from the UPS for non-HVAC (heating, ventilating, and air conditioning) purpose. This includes not only the power drawn from actual machines or network switches, but also the loss from the power distribution units (PDUs) or power supply units (PSUs). According to the investigation on two different types of datacenters, each datacenter demonstrated rather different characteristics in power usage. For datacenter facility 8.1, 54% of its available power was consumed in the HVAC while only 38% was for computer loads. In contrast, the datacenter facility 8.2 spent the majority of its power, 63%, in the computer loads and only 23% for the HVAC. One reason for the difference is that the HVAC of facility 8.1 was running on its full power regardless of the utilization of their computing nodes. In other words, facility 8.1 will continue to dissipate power for the HVAC even if the computer loads are low. Other potential reasons for facility 8.2's higher power efficiency for computing, although not revealed in the original report, could be attributed to different ambient temperatures, different sizes of the facilities, different designs of air flow, etc.

To emphasize the importance of efficient HVAC for maintaining a datacenter, we can perform simple math to see what if the datacenter 8.1 could achieve the HVAC efficiency of datacenter 8.2. If the datacenter 8.1 can reduce its power consumed by the HVAC down to 23% of its total power as in the datacenter 8.2, the amount of power for the HVAC will be reduced from 312kW ($\sim 578kW \times 0.54$) to 79kW ($\sim 578kW \times 0.46 \times \frac{23}{100-23}$). The difference in power, 233kW, will become 2041MWH ($\sim 233kW \times 365days \times 24hours$) per year. When

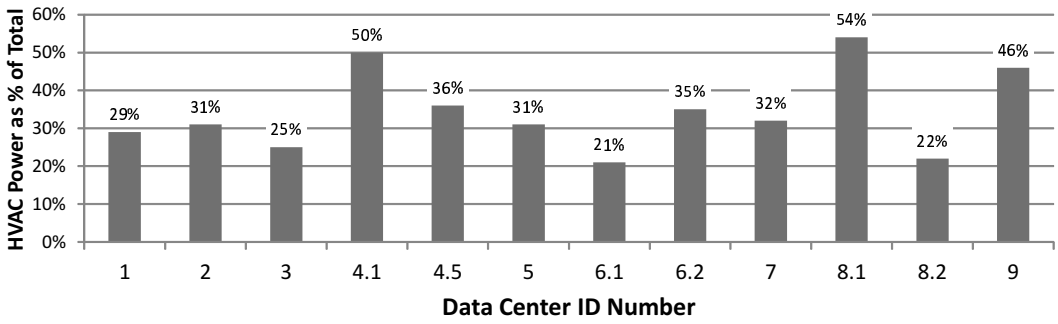


Figure 7: Power consumed by HVAC out of total power in datacenters [27]

this energy saving is converted into dollars by applying roughly $0.1/1kWH, 2041MWH will turn into 204 thousand dollars. This shows why power-efficient cooling is critical in datacenters. Figure 7 shows the power consumed by the HVAC for a variety of datacenters investigated in [27], the power portion by the HVAC alone spans from as low as 20% up to more than 50%.

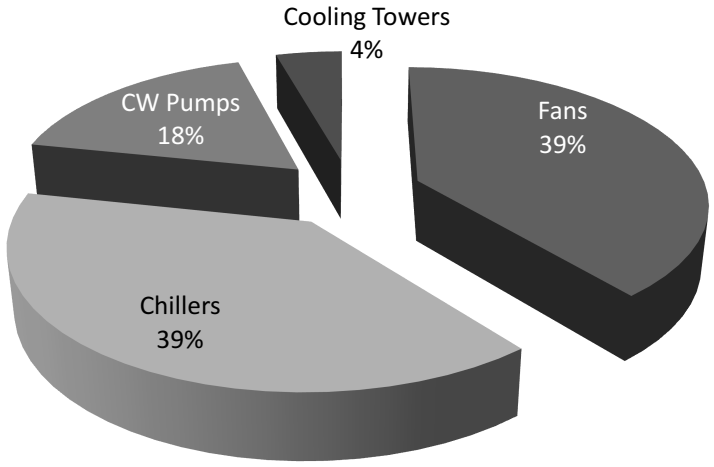


Figure 8: HVAC power breakdown

Furthermore, the power breakdown of the HVAC itself for datacenter 8.2 is shown in Figure 8 based on data collected in [3]. According to this study, there are three major power consumers in an HVAC: fans (39%), chillers (39%), and cooling water pumps (18%). In a typical datacenter with raised floor, fans designed to circulate the air in the server room

are connected to two water pipes, one for inlet of cool water and the other one for outlet of warm water. Pumps in Figure 8 are for the water flow while chillers are for cooling down the warm water.

3.2 Mathematical Modeling of Performance and Utility Consumption for A Heterogeneous Cloud Computing Environment

Cloud computing has emerged as a highly cost-effective computation paradigm for IT enterprise applications, scientific computing, and personal data management [28, 29]. Given the cloud service is to be provided by machines of various capability, performance, power, and thermal characteristics, it becomes a challenging task for providers to understand their cost effectiveness when deploying their systems. This dissertation analyzes a parallelizable task in a heterogeneous cloud infrastructure with mathematical models to evaluate the trade-off of energy and performance. To achieve the optimal performance per utility, the response time of the slowest node should be no more than three times of that of the fastest node. Theoretical analysis presented here can be used to guide allocation, deployment, and upgrades of computing nodes for optimizing utility effectiveness in cloud computing services [30].

3.2.1 Background

Cloud computing is an emerging computing paradigm that is transforming the entire IT industry, high performance computing, and even personal data sharing and management. The basic concept of cloud computing is that the computing power is supplied as flexible as utility, similar to electricity or water. As such, computing resources can be centrally managed, maintained, and upgraded by a service provider, offloading the burden of small business owners or those who do not have expertise or budget to handle the fast-changing computing infrastructure. Nevertheless, the core idea of cloud computing is not completely new; it has been evolved from previous legacy systems — for example, grid computing, clusters, or autonomic computing. To differentiate cloud computing from grid computing,

several characteristics of the cloud were identified [31]. First, cloud computing makes use of virtualization techniques to isolate users from the physical resources [31]. Second, cloud computing offers more flexibility than the legacy infrastructures in terms of pricing and dynamic scalability. Last but not least, the performance of computing in the cloud is guaranteed through service level agreement (SLA), rather than deterministic or dedicated physical resources [32].

In addition to IT enterprise applications and personal data management, there is also a growing interest in performing high-performance computing (HPC) in the cloud [33]. This paradigm shift can substantially reduce the total cost of ownership by eliminating the need of maintaining large-scale parallel machines and their enormous power and cooling system [34]. From the perspective of cost-effectiveness, there are trade-offs in terms of resource provisioning given that a target task can be embarrassingly parallelized, a common case for throughput-oriented computing. For example, assume that an HPC job, which can be perfectly parallelized, takes eight hours to complete using one computing node. If the cloud computing service provider charges a job on a per (machine·hour) basis, *i.e.,* utility based on the accumulated machine time, instead of running it on one node for eight hours, the job can be finished in one hour on eight machines with 8x speedup with the same utility charge (8 machine·hour). In this case, (execution time)×(machine·hour) becomes 8x better when compared to the case of using only one computing node.

One trend that complicates the above trade-off is the heterogeneity in a cloud computing environment. Although a cloud service provider can start with their business with (near-)homogeneous computing nodes, it is likely that the facility will grow more heterogeneously over time due to upgrades and replacement. Therefore, not only do the performance and capability of each computing node continue to deviate, the new computing nodes will also provide better performance at the same power of the older ones due to technology scaling and architectural innovation. Due to this heterogeneity, there will be significant variations with respect to the response time depending on provisioning policies.

To mitigate this variation and guarantee quality-of-service, the cloud provider may want to dismiss the slowest computing nodes. The question to answer here is that *how slow a physical node can be for a given task to maintain its optimal computing quality in terms of execution time and energy cost?* To tackle the issue, this section establishes a mathematical model based on statistics for a heterogeneous cloud environment. Using this model, this section evaluates the trade-off of execution time and energy of a task to understand optimal provisioning in a cloud.

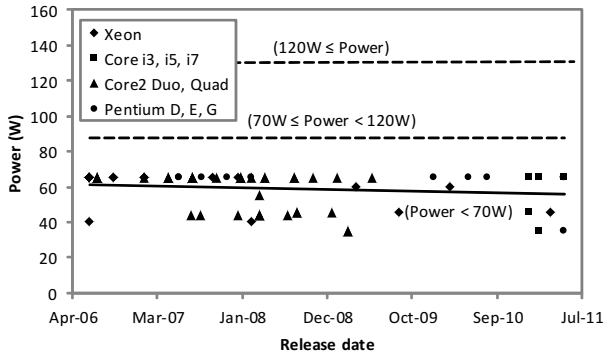
3.2.2 Cloud Computing Model

3.2.2.1 Workload definition

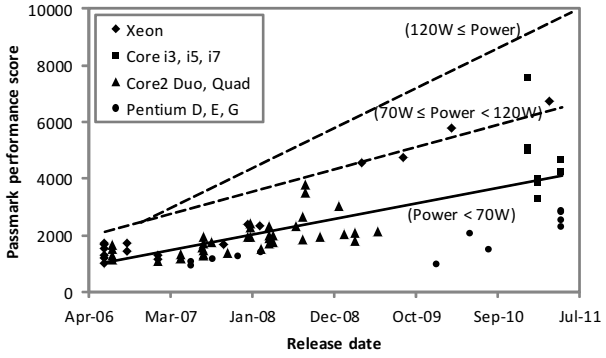
In this analytical study, the workload is assumed to be perfectly parallelizable, which is often the case for throughput-oriented computing present in HPC and transactional processing applications. For example, the most common application for cloud computing is application service on the web. For such web services, all the requests received at the same time can be processed individually and independently. Therefore, one can expect n times speedup when there are n nodes deployed if and only if the number of concurrent users is always larger than or equal to n.

Next, it is assumed that an entire workload can be evenly divided into m smaller jobs without affecting its scalability and m is also assumed to be larger than n where n represents the maximum number of virtual machines in the cloud (For simplicity, $m = kn$ where k is a positive integer.) In this study, one job unit represents the smallest task running to the end on one single physical node without interruption. However, intermittent context switches within one job unit are not considered interruption as long as the task keeps running on the same physical node.

On the other hand, a virtual machine is not allowed to be migrated among physical nodes during the execution of a job unit because this migration will not only include the executable image but also all the architectural states including memory footprint. Data migration on interconnected cloud computing nodes will likely cause significant performance



(a) Power consumption



(b) Performance

Figure 9: Power consumption and performance of Intel's CPUs since 2006 (Solid line: $Power < 70W$)

degradation due to peer-to-peer communication.

3.2.2.2 Power and performance behavior of a cloud

Before detailing the definition of the power and performance in a heterogeneous cloud, we start with the following scenario from the perspective of the cloud administrator. Typically, cloud service providers would commence their cloud computing business with a number of (near-)homogeneous computing nodes. Over time, the cloud provider will phase out some of the old computing nodes and replace them with newer nodes featuring latest technologies. Gradually, the capability and performance of all machines in the cloud will become more heterogeneous. Although prior studies had considered heterogeneity at the micro-architectural level [35] and system level [36], they all assumed heterogeneity in the same

generation of manufacturing technology. In contrast, this section considers computing heterogeneity in a broader sense.

Now we review the power and performance trend of commercial microprocessors for the last few years and use our observations as a justification for our model assumption. We first plot the thermal design power (TDP) numbers and the performance scores of PassMark [37] for different processors including Pentium, Core 2, Core i3/5/7, and Xeon under 70W since 2006.[1] The solid line shows their asymptotical trends between 2006 and 2010. In addition to this, we also plot the trends of two other machine groups in the same figure (plotted in dashed lines without individual dots) based on their TDP: 70W to 120W and over 120W.

To observe the trends, we applied regression method to estimate the relationship between power and performance over time. By taking all the samples into account, our regression models for power and performance are plotted by solid lines in these figures. As shown in Figure 9b the performance continues to improve for each machine group across different proliferations or generations. On the other hand, the TDP trend in Figure 9a shows negligible growth. More interestingly, the TDP trends for the two lower power machine classes are, in fact, decreasing. It is the consequence of recent awareness of power wall, which gradually increases the cost for heat dissipation. For the same reason, we anticipate that the power grade of future processors will remain below the bar. It also implies that with the same power budget, newer machines can deliver higher performance. In other words, performance per power continues to grow over time. For example, 95W Core i7 (Lynnfield) released in September 2009 achieves higher performance than the 95W Pentium D (Presler) in January 2006. This is largely attributed to technological advancement in micro-architecture as well as scale-down in feature size and supply voltage.

Given these observations, we move on to define our model of power and performance

[1] They include all commercial desktop or server processors from Intel from January 2006 to February 2010 except Celeron processors and certain processors that did not report TDP or PassMark results.

for a future heterogeneous cloud by making the following assumptions. First, the computing nodes in the cloud we will analyze are heterogeneous, having different micro-architectures fabricated using different processes. Thus, the cloud provides a variety of capability and performance. Second, the performance capabilities of these computing nodes are uniformly distributed (from low to high) while consuming exactly the same amount of power. The rationale behind this assumption is two-fold. First, for a given power budget, Figure 9 shows the trends of power consumption and performance for three different processor groups classified by the thermal design power. They all show that, for a given power budget, the performance of each machine class continues to improve linearly while their power envelope remains pretty much unchanged. In other words, the power efficiency measured by performance per power improves over time. Second, when a datacenter phases out some computing nodes due to upgrade, new computing nodes can safely be deployed only when the new, aggregated power consumption with these upgrades does not exceed the original one. Otherwise, the datacenter must also upgrade their power delivery infrastructure as well as the cooling capacity for accommodating the new servers. Given this overhead, we anticipate that the replacement and upgrade will be done without altering the power delivery infrastructure. Therefore, we assume that the newly deployed servers will improve performance linearly across different machine proliferations while using the same amount power. To express this distribution mathematically, we assume that the response time for executing a job unit in such cloud is uniformly distributed from a seconds (the fastest node) to b seconds (the slowest node). Hence, the probability distribution function (PDF) of the response time for executing a job unit in this cloud can be illustrated in Figure 10.

On the other hand, we assume that the cloud service provider can improve the *worst-case response time* when they dismiss physical nodes with the least performance. For example, when the cloud service provider decides to retire one third of their physical nodes from the slowest batch, we assume that the new response time for executing a job unit of

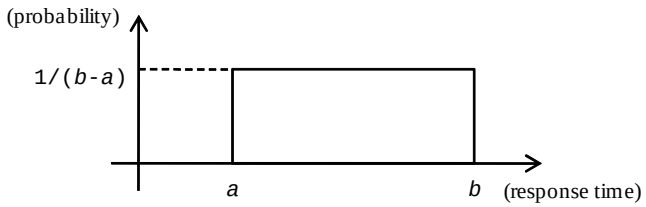


Figure 10: PDF of the execution time of a job unit when there are n virtual machines

this cloud becomes a uniform distribution from a seconds to $(a+2b)/3$ seconds, represented by $U(a, \frac{a+2b}{3})$. As such, we assume that the maximum number of virtual machines that can be allocated on this cloud also shrinks in the same ratio. In Figure 11, the impact of retiring one third of its physical nodes from the cloud is illustrated. The variable p in this figure represents the maximum number of virtual machines that can be allocated on the cloud, while n represents that of the original cloud discussed in the Figure 10. Moreover, the PDF in Figure 11 shows the improved worst-case response time as a result of removal of one-third of physical nodes from the slowest side.

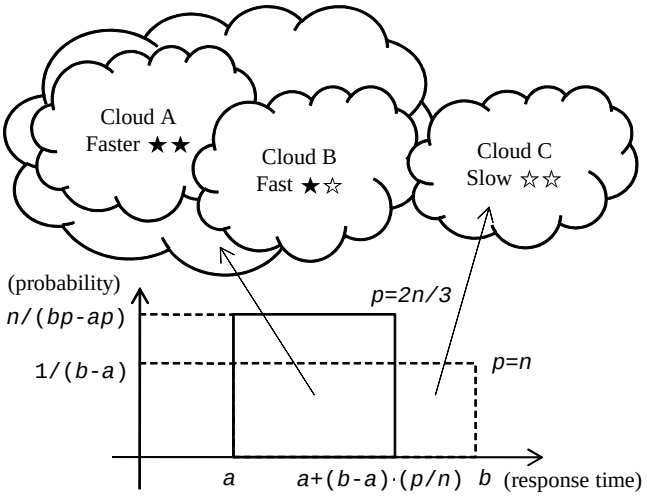


Figure 11: PDF of the execution time of a job unit when there are $2n/3$ virtual machines

Nevertheless, in the given PDF of the response time, we did not assume that a particular virtual machine can pick a physical node at a particular speed. Rather, when a probability distribution function of a cloud is given, the behavior of a virtual machine in this cloud is considered to follow the PDF in a statistical manner. In other words, we assumed that

virtual machines will be uniformly distributed across the physical nodes. Even though dispatching more jobs to newly deployed servers with higher power efficiency will lead to better energy efficiency, this is not the case for a datacenter due to the following reasons. First, for a datacenter, it is important to balance the power draw across the AC phases [16]. The balance will break when jobs are biasedly distributed to only certain computing racks. Second, it is desirable to minimize the number of hot spots for a datacenter, a common consequence of unbalanced workloads. Hot spots generally cause higher machine failure rate and require additional attention and effort for removing the heat.

3.2.2.3 Execution time and energy consumption

First, we would like to clarify the execution time of a given workload on a cloud used in this study. It is defined as the time consumed to finish the entire workload consisting of m job units. When a partial number of job units are assigned to more than one virtual machine, the execution time, in our definition, is bounded by the virtual machine that finishes the last. For example, when an animator renders a movie composed of m independent frames, the movie cannot be released before the last frame finishes rendering. In addition, when comparing the performance of cloud configurations, we assume that the *baseline* is the case of executing the same amount of workload on a virtual machine running on the fastest node. When more virtual machines are used to execute the workload in parallel, more slow nodes will be used to accomplish the task. As a result, the parallelized version could reduce the overall effectiveness of utility consumed in the cloud.

Second, we clarify *energy consumption* to be the total energy needed to complete a given amount of workload. In particular, when some physical nodes finish their assigned job units before the others, we assume that these nodes will not consume energy while waiting for the other nodes to finish. This is because, in the real world scenario, these nodes will either be assigned for other useful tasks or moved to the near-zero power state [20] for saving energy. In addition, given each computing node consumes the same amount of power, energy consumption as defined will be proportional to the total execution time. Therefore,

for a parallelized workload, its utility consumption is calculated as the summation of the execution time of each virtual machine.

To quantify the effectiveness of resource provisioning in a cloud, we use the well-known metric — energy-delay product [38] calculated by multiplying the execution time (seconds) with the energy consumption (joules). This metric will be used in our subsequent evaluation when provisioning resources (*i.e.,* the number of virtual machines should be allocated for achieving the optimal energy efficiency).

3.2.3 Analytical Evaluation

Based on the above assumptions, we now use analytical models to perform our evaluation. The evaluation will compare energy-delay product (EDP) of each configuration over the EDP of the baseline. The baseline of this study is the case of using only one virtual machine running on the fastest physical node.

3.2.3.1 The Baseline

The baseline of this study assumes that the entire job is performed on one virtual machine which is running on the fastest physical node. In this case, the fastest physical node can retire a job unit in every a seconds. Since there are m independent job units in the entire workload, the baseline configuration takes ma seconds to finish. On the other hand, this configuration consumes $W \cdot ma$ joules for completing the entire workload where W represents the power of a physical node. To sum up, the energy-delay product of the baseline of this study will be as follows.

$$EDP_{base} = (W \cdot ma)(ma) = Wm^2a^2 \tag{2}$$

3.2.3.2 Expectation-based analysis

We now analyze the execution time and energy consumption of a cloud model in an expectation-based analysis. In order to understand the expected performance, we will first discuss a new distribution function which represents the execution time of a virtual machine with more than one job unit.

Execution time distribution across virtual machines: The PDF of the response time when using p virtual machines is given by $U(a,\ a + \frac{(b-a)p}{n})$ as illustrated in Figure 11. However, when a virtual machine is responsible for more than one job unit (*i.e.*, m/p units), the total execution time of this virtual machine cannot be modeled in the same way. Rather, it can be modeled as the summation of independently chosen m/p samples from Figure 11. When we add independent samples from a uniform distribution, the distribution function of this summation tends to approach a normal distribution according to the *central limit theorem* [39]. The central limit theorem proves that when we add more independent samples into the summation, the distribution of this summation will become more like a normal distribution. In addition, the summation of twelve samples is known to be good enough to satisfy the central limit theorem [39]. In this case, we assume that a virtual machine will be responsible for more than 12 job units by letting $m \geq 12n$ (*i.e.*, $m \geq 12p$ since $p \leq n$).

Now our goal is to obtain the mean and variance of the normal distribution which represents the total execution time of a virtual machine responsible for m/p job units. First, we need to calculate the mean and variance for the original uniform distribution, $U(a,\ a + \frac{(b-a)p}{n})$.

$$\begin{aligned} \text{Mean} &= \frac{1}{2}(a + a + \frac{(b-a)p}{n}) = a + \frac{(b-a)p}{2n} \\ \text{Variance} &= \frac{1}{12}(a + \frac{(b-a)p}{n} - a)^2 = (\frac{(b-a)p}{\sqrt{12}n})^2 \end{aligned} \tag{3}$$

The central limit theorem shows that the summation of m/p independent samples from this distribution will become a normal distribution with the following mean and variance.

$$N(\frac{m}{p}(a + \frac{(b-a)p}{2n}), (\sqrt{\frac{m}{p}} \cdot \frac{(b-a)p}{\sqrt{12}n})^2) = N(\mu, \sigma^2) \tag{4}$$

For convenience, we use μ and σ^2 to denote the mean and variance of this distribution. All in all, when using p virtual machines, the execution time of each virtual machine will follow the normal distribution, $N(\mu, \sigma^2)$. The ultimate question is "how many seconds will it take for finishing the entire workload?" To answer this question, we need to first answer "what is the expectation of the largest sample from $N(\mu, \sigma^2)$ when we have to pick p samples?"

Because the overall execution time is dependent on the slowest virtual machine that finishes the last, the largest of p samples will give the total execution time. In the next section, we will use a statistical approach to answer this question.

Expectation of the largest sample: Before finding the expectation of the largest sample, we discuss the same question for the standard normal distribution, $N(0,1)$. Let $pdf(x)$ be the the PDF of the standard normal distribution. In this PDF, let y be the largest sample among randomly chosen p samples. For each case out of p cases, the probability for y to be the largest sample will be given by the following equation.

$$\text{Probability} = pdf(y) \cdot (\int_{-\infty}^{y} pdf(x)dx)^{p-1} \tag{5}$$

The expectation of the variable y is given in Equation (6).

$$\int_{-\infty}^{\infty} p \cdot y \cdot pdf(y) \cdot (\int_{-\infty}^{y} pdf(x)dx)^{p-1}dy = ExB(p) \tag{6}$$

For convenience, $ExB(p)$ denotes the expectation of the largest sample among p samples from the standard normal distribution. In addition, by substituting $pdf(x)$ of Equation (6) by Equation (7), the numerical values of $ExB(p)$ for various p can be obtained. We show the results in the middle column of Table 1.

$$pdf(x) = \frac{1}{\sqrt{2\pi}} exp(-x^2/2) \tag{7}$$

Since the complexity of Equation (6) grows exponentially as p increases, it is infeasible to find the exact numerical values of $ExB(p)$ for $p > 64$. To address this shortcoming, we propose a more scalable way of approximating the Table 1. This scalable solution starts from implementing a random number generator which produces random numbers from the standard normal distribution. By using this random number generator, our solution will pick p independent random samples and remember the largest sample among them. The solution will repeat this operation for a long enough time and take an average of the largest samples. This experimental way is able to generate the exact numerical values of $ExB(p)$ as shown in the rightmost column of Table 1 after averaging more than 100 million trials.

p	*By Equation (6)*	*By experimental way*
1	0.00000	-0.00001
2	0.56419	0.56419
4	1.02938	1.02938
8	1.42360	1.42356
16	1.76599	1.76591
32	2.06967	2.06968
64	2.34373	2.34368
128		2.59461
256		2.82679
512		3.04392

Table 1: Expectation of the biggest sample ($ExB(p)$) from $N(0,1)$

When comparing the middle and the rightmost column of Table 1, one can find that the mathematical accuracy is slightly compromised in exchange of the scalability. However, we do not expect the tiny error of the numbers to affect our analysis and conclusion.

The study of the largest sample in the standard normal distribution gives us a keen idea about the $ExB(p)$ for other normal distributions. Let a random variable X follows $N(\mu, \sigma^2)$ with $\mu \neq 0, \sigma \neq 1, \sigma \neq 0$ and a derived random variable $Y = (X - \mu)/\sigma$. Then Y follows $N(0,1)$ by recalling the property that if X follows $N(\mu, \sigma^2)$ and a and b are real numbers then $aX + b$ follows $N(a\mu + b, (a\sigma)^2)$. From Equation (6), the expectation of the largest sample for Y is as follows.

$$\text{Expectation of the largest sample for Y} = ExB(p) \tag{8}$$

Since $Y = (X - \mu)/\sigma$, $X = Y\sigma + \mu$.

$$\text{Expectation of the largest sample for X} = ExB(p) \cdot \sigma + \mu \tag{9}$$

Now, the expectation of the largest sample can be calculated for any arbitrary normal distribution.

Execution time and energy consumption analysis: In our model, each one of the p virtual machines is responsible for m/p job units, and the response time for each job unit follows $U(a,\ a + \frac{(b-a)p}{n})$. Now, the expectation of the time required on a virtual machine

finishing the last can be calculated by the conclusion from the previous sections.

$$\begin{aligned}\text{Execution time} &= \mu + ExB(p) \cdot \sigma \\ &= \frac{m}{p}(a + \frac{(b-a)p}{2n}) + ExB(p) \cdot \sqrt{\frac{m}{p}} \cdot \frac{(b-a)p}{\sqrt{12}n} \\ &= \frac{ma}{2np}(2n + (\frac{b}{a} - 1)p + ExB(p) \cdot p^{3/2} \sqrt{\frac{1}{3m}} \cdot (\frac{b}{a} - 1)) \\ &= \frac{ma}{2np}(2n + (\frac{b}{a} - 1)p + Unbalance(\frac{b}{a}, p, m))\end{aligned} \tag{10}$$

In this equation, we name the second term *Unbalance*, which becomes zero if and only if every single virtual machine finishes at the same time.

$$Unbalance(\frac{b}{a}, p, m) = ExB(p) \cdot p^{3/2} \sqrt{\frac{1}{3m}} \cdot (\frac{b}{a} - 1) \tag{11}$$

For example, a higher deviation from the normal distribution indicates that the random samples from this distribution are more spread out, which increases the probability of having more deviated samples. In our case, since the finishing time of a virtual machine is modeled by picking a sample from Equation (4), more deviated samples indicate that the workload assignment is unbalanced among virtual machines executing this workload. In particular, a larger (b/a) will lead to a larger σ^2 in Equation (4) and a larger $Unbalance(\frac{b}{a}, p, m)$ in Equation (11). Hence, we can conclude that a larger (b/a) value causes more unbalanced workload distribution among virtual machines, degrading the overall performance. Also note that $Unbalance(\frac{b}{a}, p, m)$ is directly proportional to $1/\sqrt{m}$. Since m is independent of p or b/a, changing the value of m will not affect other variables in Equation (11). This implies that a very large m will eventually zero out Equation (11). Thus, the execution time when $m \to \infty$ can be expressed as follows.

$$\text{Execution time (when } m \to \infty) = \frac{ma}{2np}(2n + (\frac{b}{a} - 1)p) \tag{12}$$

Meanwhile, the energy consumption has to be evaluated probabilistically as well. As we defined that the performance is bounded by the execution time of a virtual machine that finishes the last, the expectation of the largest sample from Equation (4) needs to be calculated. In contrast, for evaluating the utility consumption, we need to focus on the average

execution time of p virtual machines. This is because, in a normal distribution, the probability for having $\mu + \alpha$ samples is exactly the same as having $\mu - \alpha$ samples. This fact indicates that the odds of having a virtual machine consuming α seconds more than the average is the same as having a virtual machine consuming α seconds less than the average. Therefore, we can conclude that the expectation of the total execution time is given by μ times p, the number of virtual machines. Given the power of a physical node in the cloud to be W, the total energy consumption will be the following.

$$\begin{aligned} \text{Energy consumption} &= W \cdot \frac{m}{p}(a + \frac{(b-a)p}{2n}) \cdot p \\ &= W \cdot m(a + \frac{(b-a)p}{2n}) \end{aligned} \tag{13}$$

$$\begin{aligned} EDP_{exp}(p) =& \frac{Wm^2a^2}{4n^2p} \cdot ((2n + (\frac{b}{a} - 1)p + Unbalance(\frac{b}{a}, p, m))(2n + (\frac{b}{a} - 1)p)) \\ =& \frac{EDP_{base}}{4n^2p} \cdot ((2n + (\frac{b}{a} - 1)p + Unbalance(\frac{b}{a}, p, m))(2n + (\frac{b}{a} - 1)p)) \end{aligned} \tag{14}$$

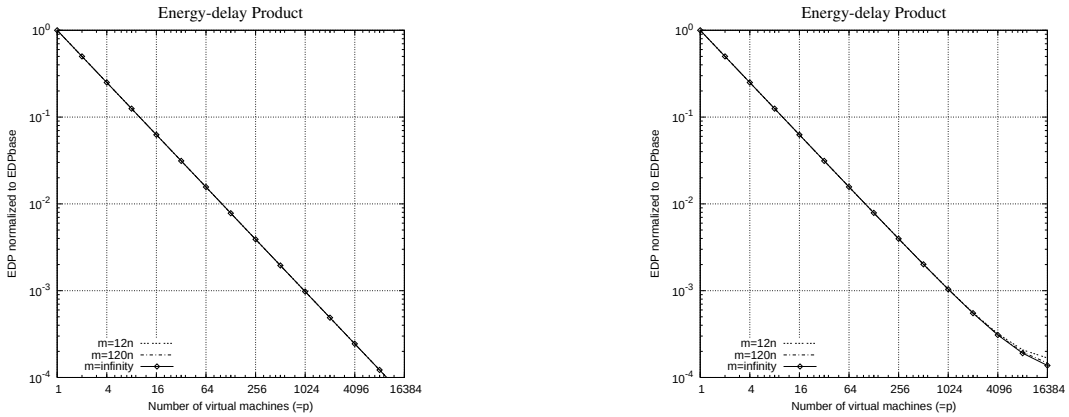
Similarly, the energy-delay product for $m \to \infty$ can be calculated as follows.

$$EDP_{exp,m\to\infty}(p) = \frac{EDP_{base}}{4n^2p} \cdot (2n + (\frac{b}{a} - 1)p)^2 \tag{15}$$

To visualize the effect of a large m in the EDP_{exp} metric, $m = 12n, m = 120n$, and $m \to \infty$ are illustrated in Figure 12 by using the following coefficients; $n = 16384$, $b/a = 1, 2, 3, 5$, and $ExB(p)$ from Table 1. To find the exact value p that makes the EDP metric be a global minimum point, we take the derivative of Equation (15) with respect to p and set it zero.

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}p}(EDP_{base} \cdot \frac{(2n + (\frac{b}{a} - 1)p)^2}{4n^2p}) &= 0 \\ p &= \frac{2n}{\frac{b}{a} - 1} \ (\because p > 0) \end{aligned} \tag{16}$$

In the example of $m \to \infty$ in Figure 12, the minimum EDP is achieved when $p = \frac{2n}{b/a-1} = 16384$ in Figure 12c or $p = \frac{2n}{b/a-1} = 8192$ in Figure 12d.



(a) $b/a = 1$ (b) $b/a = 2$

(c) $b/a = 3$ (d) $b/a = 5$

Figure 12: An example of the expectation based analysis where the total number of available virtual machines is 16384

Again, $p = n$ has to be fulfilled while maintaining Equation (16) to be energy-effective for all n virtual machines in the cloud. By combining two conditions, $p = n$ and Equation (16), the requirement of b/a can be calculated as follows.

$$\begin{aligned} n &= \frac{2n}{\frac{b}{a} - 1} \\ \frac{b}{a} &= 3 \end{aligned} \tag{17}$$

This equation suggests that in a heterogeneous cloud computing environment with uniformly distributed performance, physical nodes that respond 3x slower than the fastest one should not be used when the provisioning objective is to minimize the EDP.

3.3 SimWare: A Holistic Datacenter Simulator

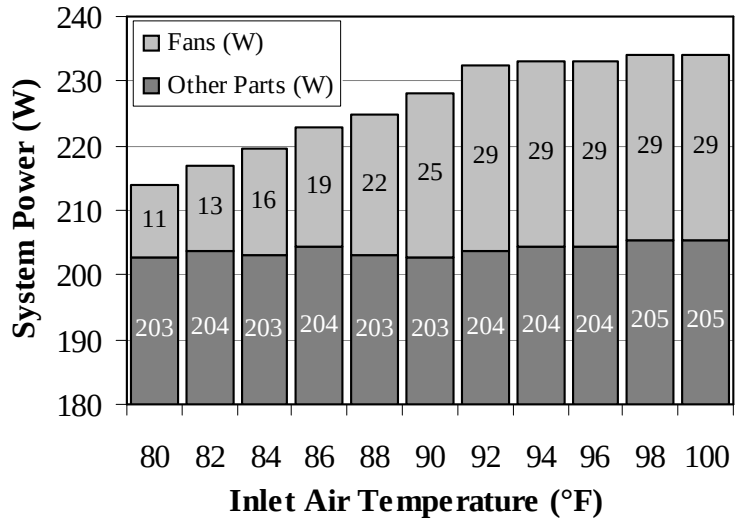
3.3.1 Background

One critical missing part of previous studies [7, 8, 9, 10, 11, 12] is the ignorance of the temperature dependency on server power. In general, a server operating at a higher temperature consumes more power than a server at a lower temperature. To briefly show the relationship, an experiment using a Xeon 5160 system is performed. While running LINPACK benchmark to keep the processor fully loaded, the entire system power, fan power, fan speed, and core temperature at different inlet-air temperatures are measured. The entire system power versus the inlet-air temperature is depicted in Figure 13a. Clearly, the system power increases as the inlet-air temperature increases with a major contribution of increased fan power. As illustrated in Figure 13b, the fan speed steeply increases whereas the temperature of the processor remains the same until the inlet air reaches $92°F$. In short, servers consume more power under high temperature than servers under low temperature primarily due to increased fan power.[2]

The following hypothesis rationalizes the increased fan power in Figure 13a. It is first assumed that a core temperature is $70°C$. When the inlet-air temperature is $10°C$, the temperature difference between them is $60°C$. However, when the surrounding temperature is $40°C$, the temperature difference becomes $30°C$. As a result, the latter requires twice more air than the former and the fan must rotate twice faster. Hence, power consumption of server fans increases as the inlet-air temperature increases.

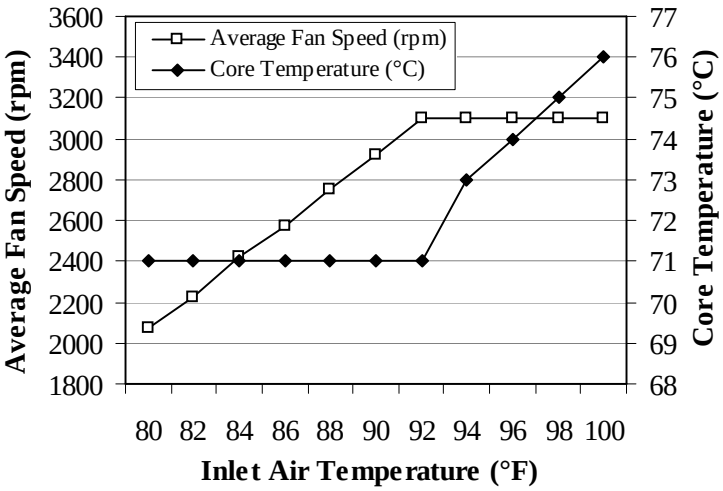
Prior studies ignored changes in fan power, which accounts for 10-30% of the total system power [20]. Assuming constant fan power will result in too optimistic results. Many of these proposed techniques for saving cooling energy leave servers at higher inlet-air temperature than the baseline [9, 10, 11]. They may save significant amount of energy in cooling units; however, their implications to the server power should be evaluated carefully

[2]The proposed research ignores data points over $92°F$ ($\sim 33.3°C$) as fans reach their maximum speed and core temperature start to diverge. In addition, this temperature range is over emergency temperature of A1 class server ($32°C$) [40]



(Other Parts: CPU, Memory, HDD, Motherboard, etc.)

(a) Power draw.



(b) Fan speed and core temperature.

Figure 13: Inlet-air temperature versus system power, fan power, core temperature, and fan speed.

by taking all the components into account.

Moreover, previous studies [9, 10, 11] disregarded the travel time of the cool air flowing from the computer room air conditioning (CRAC) units to servers. Above all, energy efficiency should be achieved in accordance with that the inlet-air temperature ($T_{inlet\ air}$) of

servers remains below the emergency temperature ($T_{emergency}$). If no travel time from the CRAC units to the servers is considered, the datacenter can easily maximize its power savings by setting the CRAC units to raise the supply-air temperature until one of the servers reaches the emergency temperature ($\forall T_{inlet\ air} = T_{emergency}$). However, in reality, when the CRAC units detect that $T_{inlet\ air} > T_{emergency}$ for a server and start to supply cool air to lower $T_{inlet\ air}$ below $T_{emergency}$, the server will stay above $T_{emergency}$ until cool air arrives. In other words, a time delay exists for cool air to flow from CRAC units to a server, and the server will fail to remain below $T_{emergency}$ during that period of time. To avoid such failure, CRAC units must secure a safety margin (T_{safety}) when raising the supply-air temperature. Therefore, a new simulator, SimWare, is introduced in this section. SimWare presents a method to estimate the air-travel time from CRAC units to servers and shows the amount of the cooling efficiency loss due to T_{safety}.

3.3.2 Core Components of SimWare

This section describes the building blocks, input files, and configurable parameters of the datacenter simulator, SimWare. As shown in Figure 14, the simulator supports different types of utilization traces as input files and generates performance, power and temperature related statistics. SimWare consists of server-level and datacenter-level power models. The server-level model estimates the power consumption of a server by the utilization and the inlet-air temperature. In other words, the simulator considers the thermal impact on the server power. For the datacenter-level power models, the simulator uses the concept of heat distribution matrix (HDM) [41] and a CRAC power model from other study [9]. Moreover, unlike prior studies, SimWare takes the air-travel time from CRAC units to servers into account. In addition, the simulator is ready for evaluating various job scheduling algorithms and virtual machine-related [42, 43] studies. The above building blocks are combined to construct the holistic datacenter simulator.

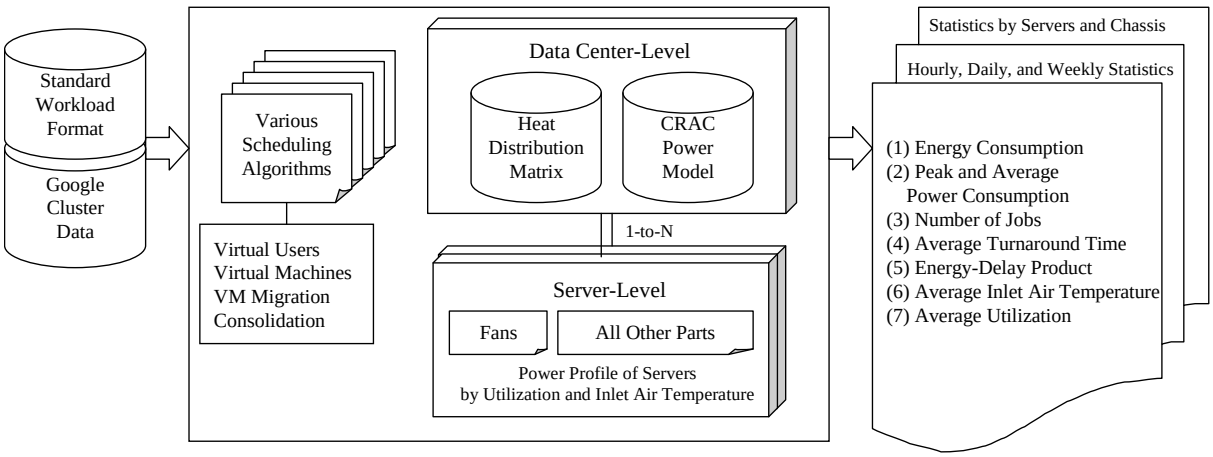


Figure 14: Overview of SimWare.

3.3.2.1 Modeling Thermal Impact on Server Power

In modeling thermal impact on the server power, SimWare relies on the laws of convective heat transfer and fan affinity laws [44]. The laws of convective heat transfer state that heat transfer (in watts) is directly proportional to the amount of air and the temperature difference between the cooling object and surrounding air. In other words,

$$Heat\ Transfer(Watts) \propto Temperature\ Difference \times Amount\ of\ Air. \tag{18}$$

For simplicity, this document assumes that the density of air is constant at the temperature range of interest. The fan affinity laws define the relationship of the rotational speed, the amount of air, and the power of the fan as

$$Amount\ of\ air \propto Fan_{RPM}, \tag{19}$$

$$Fan_{Power} \propto Fan_{RPM}^{3}. \tag{20}$$

It is first assumed that the power consumption of a CPU remains constant while the surrounding temperature increases from $T_{inlet\ air}$ to $T_{inlet\ air} + \alpha$. Meanwhile, the amount of heat transfer remains constant. When a surrounding temperature changes from $T_{inlet\ air}$ to $T_{inlet\ air} + \alpha$, the initial temperature difference ($= \Delta T$) between the CPU and the surrounding air decreases to $\Delta T - \alpha$. In Equation (18), when the temperature difference decreases

by $\frac{\Delta T-\alpha}{\Delta T}$ times, the amount of air must be increased by $\frac{\Delta T}{\Delta T-\alpha}$ times to maintain constant heat transfer. As indicated in Equation (19), to supply $\frac{\Delta T}{\Delta T-\alpha}$ times more air, the fan must rotate $\frac{\Delta T}{\Delta T-\alpha}$ times faster than before. As a result, according to Equation (20), the increased fan speed consumes $(\frac{\Delta T}{\Delta T-\alpha})^3$ times more power than when the surrounding temperature is $T_{inlet\ air}$. These laws let us calculate the relative fan power according to the power consumption of CPU and $T_{inlet\ air}$. Section 3.3.2.6 defines the boundary conditions so that SimWare can calculate the exact power consumed by the fans.

3.3.2.2 Air-Travel Time from Cooling Units

A number of factors will affect the air-travel time including the datacenter layout, the proximity of the CRAC unit to the servers, the air velocity discharged from the CRAC unit, and the height of the plenum. By considering these physical parameters, SimWare presents a simple thermodynamics-based scheme to estimate the air-travel time. Note that this scheme assumes the most optimistic scenario that will result in the fastest possible travel time. During simulation, it was found that a longer air-travel time than the most optimistic scenario worsens cooling efficiency. Therefore, to show the lower bound of the impact of the air-travel time, this document estimates the fastest possible travel time.

It is assumed that the CRAC unit discharges $8m^3/s$ of cool air into the plenum. The air fills the plenum before the tiles ($0.6m \times 0.6m$) discharge cool air. In other words, cool air fills and pressurizes the area beneath the dashed line in Figure 15 before it is supplied to the computer room. The time for this can be calculated by dividing the volume of this area by the discharging rate. In reality, after the plenum is pressurized, each tile discharges different amount of air.To simplify, the most optimistic scenario is assumed in which all tiles discharge the same amount of air.

Once the tiles supply cool air, some airflow will bypass in the direction of A and B in Figure 15, and the majority will fill up the volume above the tiles, or the cold aisle. Here, it is assumed that the supply air does not bypass but only fills the cold aisle. Altogether, the results of the calculation show that the cool air takes about six seconds to reach the

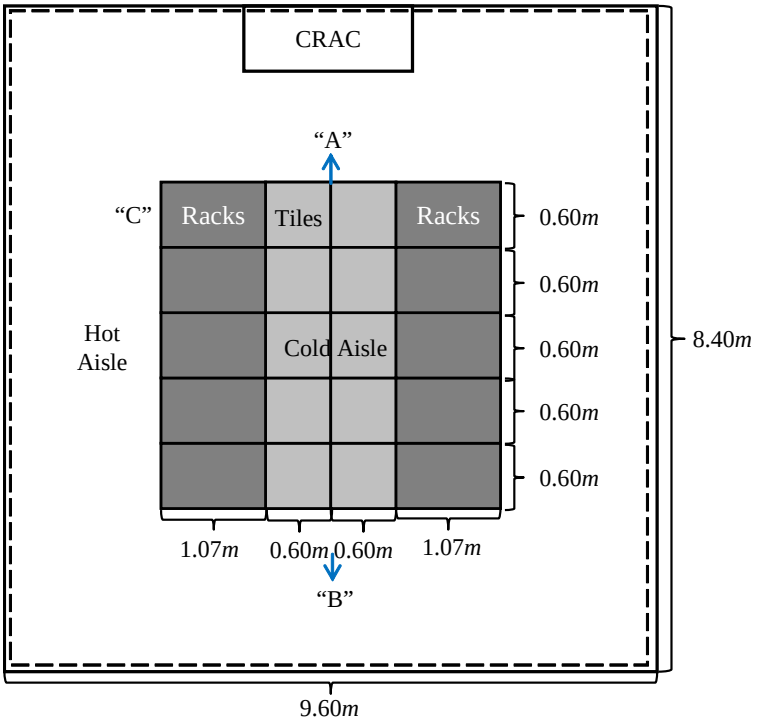


Figure 15: Layout of the raised floor datacenter.

servers located at the bottom of the rack and seven seconds to the servers at the top. In real-world scenarios, the tiles near the CRAC unit supply less cool air than the tiles far away from the CRAC unit. Certain servers such as "C" in Figure 15 are harder to cool down than the other servers. Thus, the CRAC unit usually lowers the supply temperature with a significant safety margin, thereby reducing the cooling efficiency.

3.3.2.3 Heat Distribution Matrix

In addition to the power model described previously, the heat and air flow in datacenters must also be considered. The heat and air flow can be represented by a heat distribution matrix (HDM) [41]. To the best, SimWare is the first simulator that implements heat flow, temperature, power, and performance into one single simulation infrastructure. Until now, building such a simulator has been impractical because modeling recirculated heat as workload utilization changes requires a prohibitive amount of computation. SimWare mitigates this problem by adopting HDM [41]. Generating an HDM of a datacenter requires series of

tools and simulations [45] such as computational fluid dynamics (CFD) simulations. Nevertheless, the HDM concept is simple; an HDM converts the heat generated by a particular server into an increase in temperatures of all other servers. For example, given ten servers, the size of an HDM will be 10×10. The first row of the HDM represents how much $T_{inlet\ air}$ is affected by the heat generated by the other ten servers. Matrix multiplication of the first row and the power consumption of all the servers will produce $T_{inlet\ air}$ of the first server. In other words, each cell (i, j) from the HDM indicates the contribution of server j to the temperature increase of server i. The reference datacenter has 50 blade chassis as illustrated in Figure 16. In this case, HDM of the reference datacenter becomes a 50 by 50 matrix as shown in Figure 17. For example, at the bottom-right corner, "server number 50 (from)" has tall bars for servers one through ten, indicating that the heat generated by server 50 is more likely to recirculate to servers one to ten than to the others.

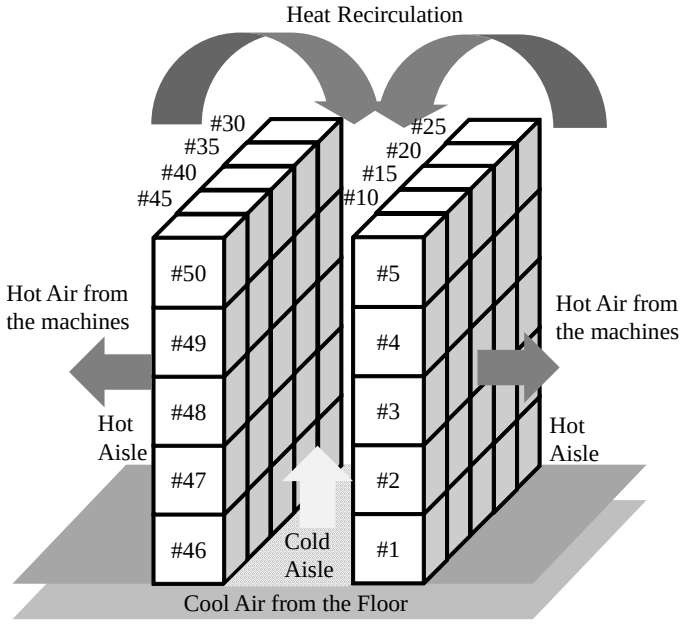


Figure 16: Simulated datacenter setup.

This heat-recirculation effect is the main reason why $T_{inlet\ air}$ varies by the location of the servers. The HDM takes this heat-recirculation effect into account and converts the impact of the power consumption (in watts) into the temperature difference (in $^{\circ}C$) of one

server on the other servers. In SimWare, an HDM is used to calculate $T_{inlet\ air}$ of every server. Since SimWare relies on HDMs as its thermal model, it inherits the limitations of HDM. Interestingly, HDM does not model changes in convective flows as a consequence of variable fan speeds; it assumes that airflow patterns are temperature invariant, which could lead to temperature estimation errors under some datacenter geometries.

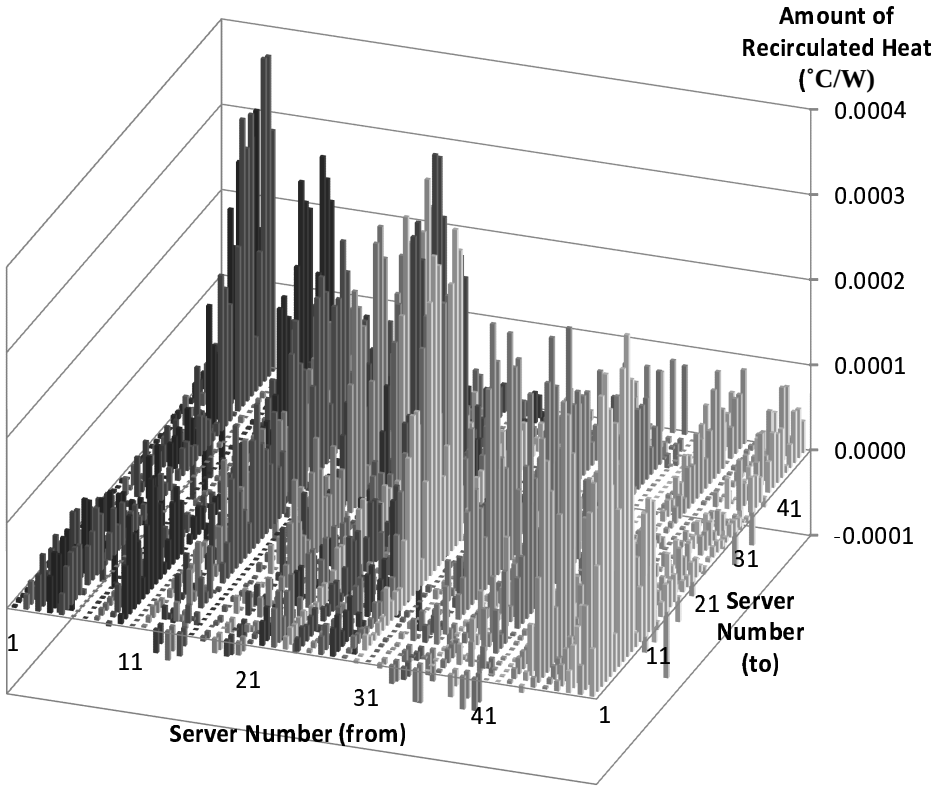


Figure 17: The heat distribution matrix used in the simulation.

3.3.2.4 Power Consumption of CRAC Units by Supply Air Temperature

Prior studies found that the power consumption of CRAC units depends on their supply-air temperature. Moore et al. [9] measured the relation between the supply-air temperature and the efficiency of the CRAC units for a typical cooling system. They showed that the power required for CRAC units can be represented as a function of the supply-air temperature and

the amount of heat that must be removed. In other words,

$$\text{Power drawn from CRAC units} = \frac{\text{Heat to remove (Power drawn from servers)}}{0.0068T^2_{supply\ air} + 0.0008T_{supply\ air} + 0.458}. \quad (21)$$

When the supply-air temperature is $10°C$, the denominator on the right-hand side of Equation (21) is about one and the CRAC units consume the same amount of power as the servers do. However, if CRAC units increase the discharging temperature ($T_{supply\ air}$), the denominator increases, and the CRAC units consume less power than when $T_{supply\ air} = 10°C$ while removing the same amount of heat. When CRAC units increase $T_{supply\ air}$ to $20°C$, they will consume only one-third of the power of the servers. In summary, SimWare uses Equation (21) to calculate the required power for CRAC units.

3.3.2.5 Input and Output of SimWare

For input traces, SimWare currently supports two formats: standard workload format (SWF) and google cluster data (GCD). A number of utilization traces in SWF collected from experimental datacenters are available in the public domain [46]. Based on ASCII, each line of an SWF file describes a submitted job and contains the job ID, the submitted time, the run time, the number of allocated processors, the average CPU time used, and the dependency between jobs. Google released GCD in November 2011, which contains similar records collected from their own warehouse-scale computers.

Once a simulation finishes, SimWare generates performance, power, energy and temperature related data including the turnaround time of the jobs for studying latency-sensitive internet datacenters [47, 48], the peak and average power consumption of servers and CRAC units, the energy usage for the given time frame, and the energy-delay product of the current configuration. Additionally, SimWare also outputs the average room temperature, the average temperature by server chassis, and the utilization level of the datacenter.

3.3.2.6 Chassis and Servers

Current simulated datacenter uses a 50 by 50 HDM. Hence, there are 50 server chassis, each holding ten blade servers to amount to a total of 500 blade servers. Each blade server

has a 130W Xeon E7-2850, one of the latest Intel products with 10 cores without Hyper-Threading. Since there are 500 servers with 10 cores each, the simulated datacenter contains a total of 5,000 cores. Despite this study is limited to 500 servers, SimWare is not limited to the current physical layout. So long as one can generate an HDM for a specific datacenter layout, SimWare can simulate it. In addition, one can use an open CFD simulator named BlueTool [45] to generate an HDM for a user-defined datacenter.

Except for the fans, the blade server consumes 260W when fully loaded and consumes half of its peak power when idle [14]. Now the specification for the fans is defined as follows. The fan on the CPU heat sink must remove heat generated by the CPU at any time. Therefore, when the fan runs at its maximum speed, it should remove 130W (the maximum CPU power) at $T_{emergency}$ (the highest operable temperature). At this operating point, it is assumed that the fan consumes 15W and runs at 3,000 rpm. Each server has two other fans with the same specification; at the front and at the back panel. The rotational speed of these case fans are directly proportional to the power consumption and $T_{inlet\ air}$ of the server. It is also assumed that fans cannot be turned off and runs at 500 rpm when the server is idle.

The emergency temperature of this server is set to 30°C ($T_{emergency}$ = 30°C), which meets A1 class server specification for datacenters [40]. Note that the goal of fan control is to save fan power and set the die temperature lower than 70°C for reliability. These numbers are close to the experiments in Figure 13 where the core temperature is at 71°C and $T_{inlet\ air}$ is measured at 92°F(~ 33°C) when the fan rotates at full speed.

3.3.2.7 CRAC Control Policy

SimWare currently supports two CRAC control policies; constant and dynamic. The constant control is the most basic strategy that a CRAC unit supplies cool air of a constant temperature. In this case, the supply-air temperature is low enough so all servers stay below the emergency temperature at any time. Because the cooling power is constant and set to the worst case scenario, this algorithm wastes cooling power when the datacenter is

under-utilized.

To tackle this inefficiency, many recent datacenters proposed dynamic CRAC control policies [49, 9]. The operation of the CRAC unit begins by supplying the lowest possible air and jumps into the main loop. In the loop, the CRAC unit gradually raises $T_{supply\ air}$ at the rate of $0.01°C/sec$ [3] until any server operates at a triggering temperature ($T_{trigger}$). When any server encounters $T_{inlet\ air} = T_{trigger}$, the CRAC unit starts to lower the supply-air temperature at the same rate, $0.01°C/sec$. In the ideal case, $T_{trigger}$ can be set as high as the emergency temperature. In such an ideal case, the CRAC unit continues to raise the supply-air temperature until any server reaches the emergency temperature. However, due to the timing delay of the CRAC units to effectively lowering $T_{inlet\ air}$, using $T_{trigger} = T_{emergency}$ as a condition will jeopardize some servers to operate unreliably above the emergency temperature. Therefore, the dynamic control policy needs a safety margin ($T_{trigger} = T_{emergency} - T_{safety\ margin}$), which leads to cooling inefficiency. The safety margin will be discussed in Section 3.3.3 after analyzing the simulation results using real-world traces.

3.3.3 Putting The Datacenter Simulator into Practice

In this section, SimWare is used to perform datacenter simulations. Among 26 available SWF files and google cluster data, the job and utilization traces from SHARCNET in 2005 is used. Results from some trace files are omitted due to their similarity. The SHARCNET utilization trace file contains about 1.2 million jobs for more than a year of operation. Figure 18 shows the daily utilization level of the simulated datacenter in a black line. From day zero to 50, the average utilization of this datacenter is less than 1%. From day 50 to 150, the workload is moderate with an average utilization of 5.3% and a maximum utilization of 44.3%. For the last phase, the datacenter is heavily used with an average utilization of 71.3%. In addition, the average power consumption from cooling units and

[3] Because the previous study [49] roughly showed that the rate is from $0.005°C$/sec to $0.015°C$/sec, $0.01°C/sec$ is used throughout this document. The rate is configurable in SimWare.

servers are also shown in Figure 18. The total power consumption generally tracks the utilization level well enough except when the datacenter is under-utilized. Because it is assumed in Section 3.3.2.6 that servers consume half of the peak power when idle, this datacenter is not energy-proportional [14]. Normalized latencies of the submitted jobs are also plotted in Figure 18. In calculating normalized latencies, the simulated latencies are compared to the latencies specified in the SWF file. Note that SHARCNET has more than 7000 cores while the simulated datacenter has 5000 cores. Therefore, normalized latencies drastically increase when the latter is at high utilization level.

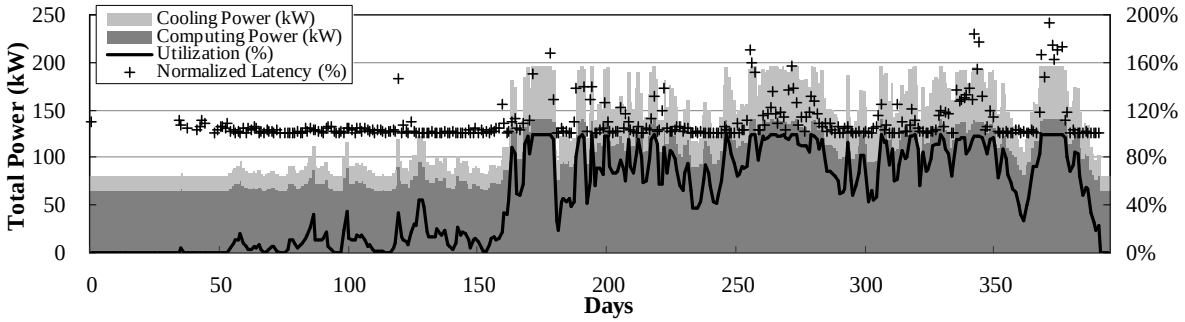
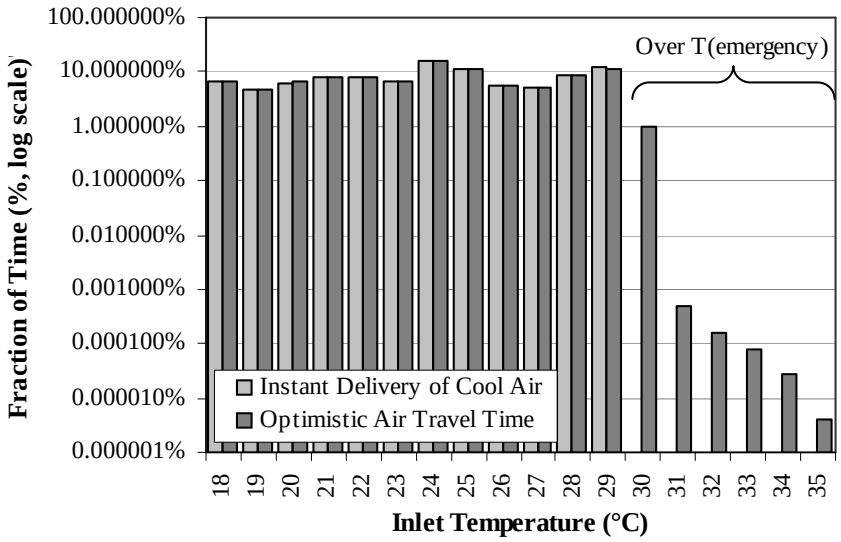


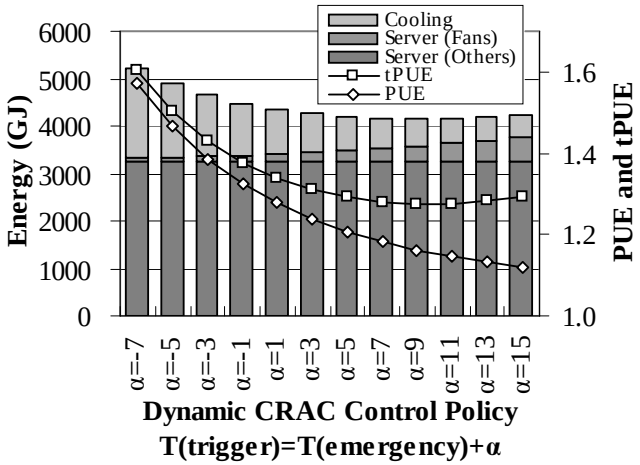
Figure 18: Utilization, latency and power trace of SHARCNET in 2005.

To demonstrate the importance of the air-travel time discussed in Section 3.3.2.2, simulations with two different configurations are considered: one with zero air-travel time by assuming that the cool air from the CRAC units instantly lowers servers' $T_{inlet\ air}$, and the second one with the optimistic air-travel time discussed in Section 3.3.2.2. These two simulations share all other parameters. As a result, the distribution of $T_{inlet\ air}$ for all the servers is depicted in Figure 19a. In Figure 19a, the Y-axis represents the fraction of time that servers spend at a given $T_{inlet\ air}$ while X-axis represents $T_{inlet\ air}$. When instant delivery of cool air is assumed, all the servers operate under the $T_{emergency}$ ($= 30°C$). However, with non-zero travel time, servers experience $T_{inlet\ air}$ over $T_{emergency}$, up to $35°C$. Therefore, to ensure $\forall T_{inlet\ air} \leq T_{emergency}$ at any time, a dynamic CRAC control scheme must secure a safety margin.

Now the safety margin ($T_{safety\ margin}$) introduced in Section 3.3.2.7 will be discussed.



(a) Effect of air-travel time.



(b) Energy breakdown and PUE.

Figure 19: Effect of air-travel time, energy breakdown, and PUE.

Even with the most optimistic air-travel time, when $T_{trigger} = T_{emergency}$, one of the servers spent more than 49% of the time at above the emergency temperature according to simulation results from SimWare. However, if $T_{trigger} = T_{emergency} - 1$, all servers will operate below the emergency temperature for 99.99% of the time. To make it 100%, $T_{trigger}$ has to be as low as $T_{trigger} = T_{emergency} - 7$. It is also found that when $T_{trigger} = T_{emergency} - 7$, the average supply-air temperature is $14.7°C$, close to the typical outlet air temperature of

the CRAC units from prior studies [4]. Figure 19b illustrates how much energy does this safety margin cost. In this figure, bars represent energy usage of the simulated datacenter. Each bar represents server and cooling energy for a given CRAC control policy. Every policy shares the same algorithm but uses different $T_{trigger}$ values. For example, the left most bar indicates that the total energy consumption is slightly more than $5{,}000GJ$ when $T_{trigger} = T_{emergency} - 7$. If two bars are compared, $\alpha = -1$ and $\alpha = -7$, the cooling energy is increased from $1100GJ$ to $1900GJ$. The safety margin costs extra $800GJ$ ($\sim 73\%$) on the cooling energy. In summary, to ensure every server to be under $T_{emergency}$ at any time, datacenters should set a safety margin, which SimWare identified as one major source of inefficiency.

α is continuously increased on the right half of Figure 19b. Along with α, the room temperature increases, and the cooling energy decreases. However, server fans now consume more energy than before, and increased fan energy now overwhelms the cooling savings. As a result, the total energy consumption saturates at $\alpha = 9$. Even though $\alpha > 9$ does not result in any energy saving, one can achieve a lower PUE than $\alpha = 9$ — an incorrect indication when evaluating energy efficiency. From $\alpha = 11$ to $\alpha = 15$, servers consume more energy and cooling units consume less than $\alpha < 11$. As a result, PUE monotonically decreases regardless of the total energy consumption. For these reasons, total PUE (tPUE) [52] is also plotted in Figure 19b. Because tPUE factors fan power out of the useful server power, smaller tPUE guarantees the better energy efficiency than bigger tPUE.

In general, the heat-recirculation effect and the air-travel time from the CRAC units result in two types of inequality among servers. Firstly, some servers will operate at relatively higher $T_{inlet\ air}$ than the others. Because hot air tends to circulate upward, the servers at the top of the racks typically experience higher $T_{inlet\ air}$ than the servers at the bottom. In simulations, the difference between the highest and the lowest $T_{inlet\ air}$ among servers is $8.1°C$. In other words, the majority of servers are over-cooled because the CRAC units lower the

[4] Prior studies reported $15.0°C$ [50, 10] or lower than $15.0°C$ [51]

supply-air temperature for the worst-case servers. Secondly, some servers require a longer time to cool down than the others. Depending on the location of the servers, $T_{inlet\ air}$ of some servers respond slowly. Because the CRAC units set a safety margin based on the worst-case scenario, these two types of inequality among servers reduce the efficiency of the cooling system and require other effective solutions.

To tackle this inefficiency, the proposed research suggests heterogeneous cooling capacities among servers for a green datacenter. If servers at the top of the racks have better cooling capacities and have higher $T_{emergency}$ than the other servers, the CRAC units can safely discharge air at a high temperature by using aggressive dynamic CRAC control policies. For example, one can pick eleven blade chassis by the highest average $T_{inlet\ air}$ from the simulated datacenter. In addition, if one change $T_{emergency}$ of these blade chassis from $30°C$ to $35°C$, the datacenter can use a dynamic CRAC control policy of $T_{trigger} = T_{emergency} - 2$ without compromising thermal guidelines and save 37% of cooling energy than the baseline, $T_{trigger} = T_{emergency} - 7$.

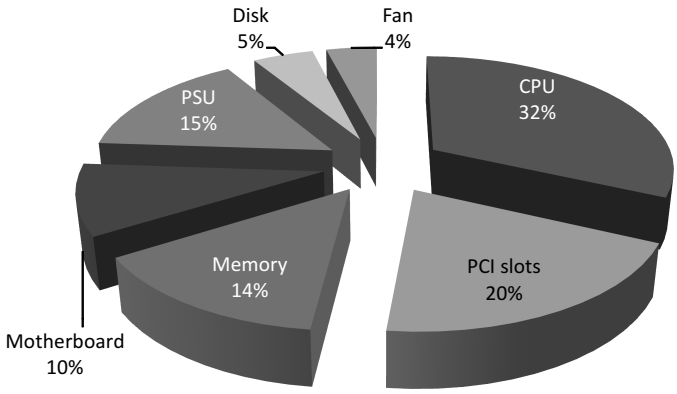
CHAPTER 4

SYSTEM-LEVEL OPTIMIZATION

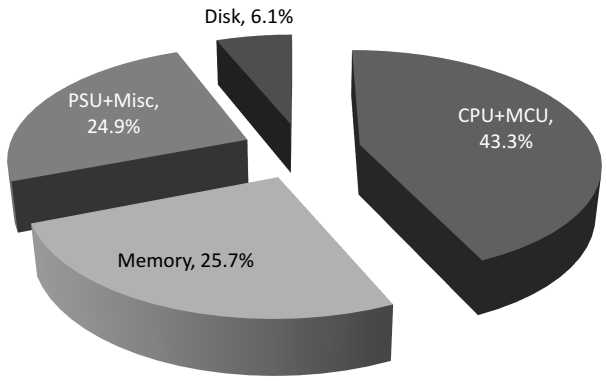
4.1 System-level Power Breakdown

Datacenter infrastructure delivers power to the components such as the CPUs, PCI slots, memory, motherboards, and disks, of a system. Before we detail the per-system power breakdown, it is important to understand why we need to estimate the actual power consumption of a system instead of the power described in a user manual, *i.e.*, the nameplate power. When calculating the nameplate power, the vendor has to be as conservative as possible to prevent their products from malfunctioning in the face of power deficiency. As a result, the total nameplate power is usually estimated by summing up the worst-case power consumption of all components in a system. In most of the cases, however, not all of the system components will operate with its maximal power simultaneously. Even if all of them are busy at the same time, a system will not reach manufacturer's nameplate power as it is oftentimes overestimated intentionally. In a datacenter environment, this discrepancy between the nameplate power and the actual measured peak power can cause significant inefficiency in the power delivery infrastructure. As seen in previous figures, a system has to be placed in a rack that typically accommodates tens of servers. Given that the power for a rack is limited by the PDU (*e.g.*, , 2.5kW per rack [53]), the number of systems in a rack is fixed based on either the nameplate power or the measured peak power of a server. For example, if a datacenter deploys servers based on the nameplate, 213W, a rack of 2.5kW will accommodate 11 servers while the actual aggregated peak power of 11 servers is less than 1.6kW [53]. Under such circumstances, the datacenter will pay more on the power delivery infrastructure for supporting the nameplate power that can never be reached.

Because the nameplate power is different from the actual power consumption, so does the power breakdown of a server is. According to the nameplate power readings in Figure 20a, a CPU accounts for around one third of the total power of a system followed by



(a) Nameplate power readings [53]



(b) Actual peak power [54]

Figure 20: Power breakdown of a server

20% for the PCI slots, 14% for the memory, and 10% for the motherboard. On the other hand, Figure 20b shows the actual power consumption of components in a typical blade server using a 2.2GHz AMD Turion processor. Different from the nameplate power readings, the CPU with an on-die MCU consumes 43% of the total actual power while the memory accounts for a quarter of the total. By comparing these two figures, it is apparent that in the actual deployment, the CPU and memory are the most power-consuming components in a system.

Figure 21 also identified that the CPU and memory are the two major power consuming

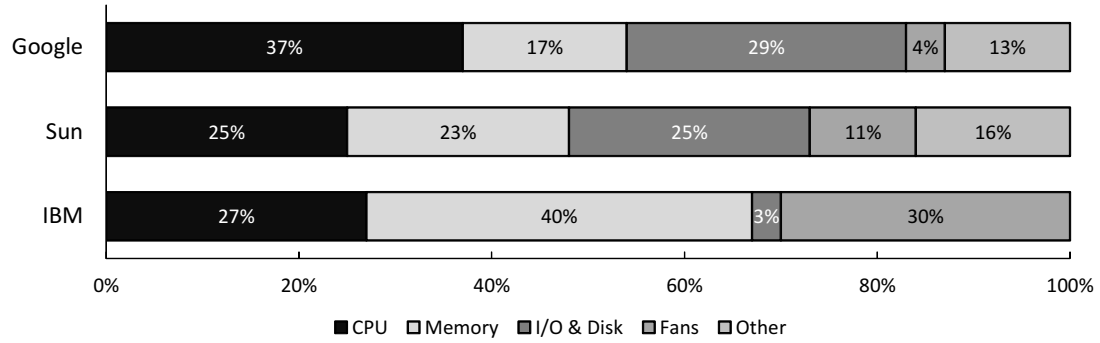


Figure 21: Per-system power breakdown by company [20]

components in a system, accounting for more than half in all three samples that corroborates the data points shown in Figure 20. In the worst case, the IBM p670, 67% of the total power were consumed by the CPU and memory. In addition to this fact, it is also interesting to find that Google spends more power on the CPU and I/O devices than the others. This is simply because their main applications are the web search, email, and document services [53]. For the web search service, many computing nodes in the back-end have to sort and index web pages while the front-end nodes have to parse queries. Many of these operations are CPU intensive. On the other hand, email services require a large number of database accesses and file downloads which are primarily I/O operations. Moreover, even though the source [53] did not mention YouTube service or similar types of workloads, it is obvious that these streaming services will demand much more on the I/O side. In summary, the most power-consuming components in a real datacenter are the CPU and memory, however, depending on the services that a system provides, the power breakdown can be vastly different.

4.2 ATAC: Ambient-Temperature-Aware Capping For Power Efficient Datacenters

The emergence of cloud computing has created a demand for more datacenters, which in turn, has led to the substantial consumption of electricity by computing systems and cooling units. Although recently built warehouse-scale datacenters can nearly completely

eliminate cooling overhead, small to medium datacenters, which still spend nearly half of their power on cooling, still labor under heavy cooling overhead. Often overlooked by the cloud computing community, these types of datacenters are not in the minority: They are responsible for more than 70% of the entire electrical power used by datacenters. Thus, to tackle the cooling inefficiencies of these datacenters, we propose ambient temperature-aware capping (ATAC), which maximizes power efficiency while minimizing overheating. ATAC senses the ambient temperature of each server and triggers a new performance capping mechanism to achieve 38% savings in cooling power and 7% savings in total power with less than 1% degradation in performance [55].

4.2.1 Background

Generally speaking, the major usage of electrical power falls into two categories: computing and cooling. Data have been shown that cooling power in a datacenter can take from 10% [4] to as much as 50% [3] of the total power depending on their operation. A metric called *Power Usage Effectiveness* (PUE) [5] as shown in Equation (22) has been widely adopted to measure the efficiency of a datacenter.

$$\text{PUE} = \frac{Power_{Servers} + Power_{Facility} + Power_{CRAC}}{Power_{Servers}} = 1 + \frac{Power_{Facility}}{Power_{Servers}} + \frac{Power_{CRAC}}{Power_{Servers}} \tag{22}$$

Given its definition, a datacenter with an ideal efficient cooling system (*i.e.,* zero cooling) will reduce the PUE value to 1. However, using PUE to evaluate the energy efficiency of an entire datacenter can be misleading. For example, it does not account for the increased fan power that consumes non-negligible power in computing servers [6]. These fans in the servers will blow harder and consume more power when a datacenter administrator reduces the cool air supply by turning down the *Computing Room Air Conditioning (CRAC)* units for power reduction. In consequence, the inlet-air temperature arises, however, the PUE value gets lowered.[1]

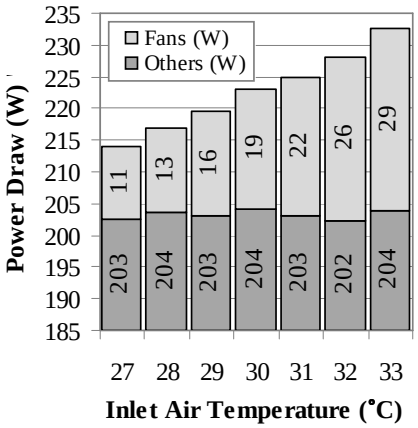
[1]With increased fan power $Power_{servers}$ will be increased, while $Power_{CRAC}$ is reduced and $Power_{Facility}$ remains constant. As a result, PUE becomes smaller.

Figure 22a shows the power breakdown of a server with a fully utilized Xeon 5160 processor. This server runs the LINPACK benchmark at different inlet-air temperatures from $27°C$ to $33°C$. As we increase the temperature, fans, which consume from 10% to 30% of the total system power [20], rotate faster and consume more power while the other parts of the server including CPU, motherboard, and disks do not show significant increase. In PUE, increased fan power is captured as part of the useful server power. Therefore, although a high ambient temperature (HTA[2]) datacenter [2] achieves a lower PUE value, it does not always guarantee a better power efficiency. To overcome such shortcomings, Hamilton [52] proposed a new metric, total PUE (tPUE) to factor fan power out of the useful server power. In this work, we will show both PUE and tPUE values in our experiments and demonstrate tPUE as a better metric in assessing power efficiency for datacenters.

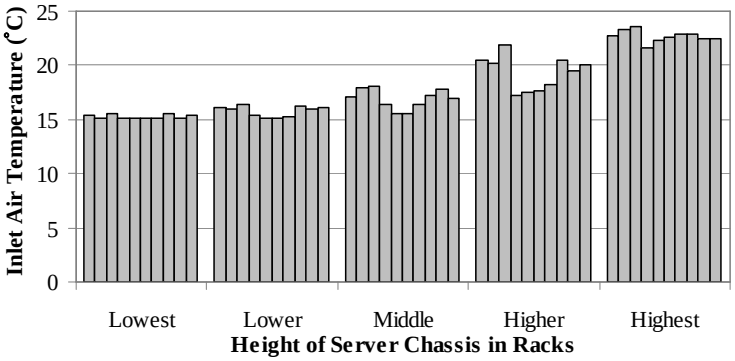
Recent study advocated the importance of taking a holistic approach when analyzing power efficiency of datacenter [12, 56]. The factors such as inlet-air temperature, power of cooling units, the effect of heat recirculation, and the impact of timing delay of cool air deliver should be simultaneously evaluated under the same simulation framework [56]. For example, the heat-recirculation effect in a datacenter results in unequal thermo-dynamic environments among servers, *i.e.,* some servers will operate at relatively higher inlet-air temperature than the others. As hot air tends to rise up, the servers at the top of racks typically experience higher inlet-air temperature. When the CRAC unit targets its cooling objective for the worst-case hot spot, those servers located at the lower level of racks are overly cooled. Such inequality was identified as the major reason of low cooling efficiency. Using SimWare, a holistic datacenter simulator in [56], we studied the temperature differential for 50 blade server chassis with 5 blade servers each. When all the blade servers are fully loaded and consume nearly 565W with the cooling units constantly blowing cool air at $15°C$, the difference between the highest and the lowest inlet-air temperature is $8.5°C$. Figure 22b details the temperature differences among servers. The left bars represent the

[2]Although the abbreviation for High Ambient Temperature is HAT, we follow Intel's naming convention.

servers closer to the bottom of racks, and right bars are those at the top. Since the CRAC unit has to keep all the servers below the emergency temperature to guarantee reliability, some server was overly cooled with an inlet temperature about 8.5°*C* below the emergency temperature.



(a) Power draw by inlet-air temperatures.



(b) Inlet temperature when CRAC supplies 15°*C* air and full server load.

Figure 22: Server power consumption by changing inlet-air temperatures.

To address cooling inequality and inefficiency, we propose a novel system-level approach called *Ambient Temperature Aware Capping* (ATAC) at per-server level for a datacenter. The technique exploits the non-uniformity of the inlet-air temperature among servers of a rack to improve the cooling effectiveness. It allows each server to run at a higher ambient temperature and applies local DVFS using its sensed inlet-air temperature as input to avoid overheating. With such dynamic regulation, the power of CRAC units

for the datacenter can be tuned down, thereby reducing the amount of cool air supply. In summary, this section makes the following contributions:

- We analyzed the energy and thermal impact of high inlet-air temperature in modern datacenters. With thorough experimentation we identified the inter-relationship for several pertinent factors including total server power, fan speed, core temperature, and fan power in response to the changes of ambient temperature.
- We proposed a new system-level technique that increases the supply air temperature of the CRAC units to optimize energy usage for the entire datacenter, while relying on a dynamic performance capping mechanism (ATAC) to keep processors from running across the emergency temperature.
- We used SimWare, a holistic datacenter simulator, to extensively study our proposed ATAC scheme and evaluate its impact to power and performance against prior power optimization techniques including Power Capping [21] and PowerNap [20].

The rest of the chapter is organized as follows. Section 4.2.2 presents the motivation of proposed scheme by showing thermal impact on server and fan powers. Section 4.2.3 discusses ATAC. Section 4.2.4 describes the simulation platform and specifies the parameters for the modeled datacenter. Section 4.2.5 evaluates and analyzes the results. Section 4.2.6 highlights the distinction of this chapter by discussing relevant research works.

4.2.2 Motivation

Server's inlet-air temperature impacts the core temperature, the server power usage, and the fan speed, which altogether creates a complex interaction among these parameters and was not properly quantified and analyzed in prior datacenter cooling literatures [57, 10, 11]. To evaluate the influence of ambient inlet-air temperature, we set up a server enclosed in a controlled area with a thermocouple and run LINPACK benchmark at the maximum load. The ambient temperature ($T_{inlet\ air}$) inside is increased due to the enclosure preventing cool air from flowing in. We repeat the experiments for three core frequencies: 2.7 GHz, 2.9

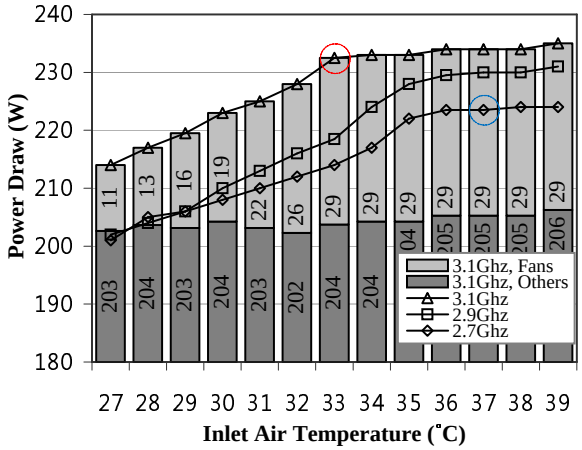


Figure 23: Inlet-air temperature versus power.

GHz, and 3.1 GHz. During the experiments, the system level power, the fan speed, and the core temperature are measured at different inlet-air temperatures, as depicted in Figure 23 through Figure 25.

4.2.2.1 Thermal Impact on Server Power

First, Figure 23 shows power consumption of the server at various $T_{inlet\ air}$. Three solid lines show system-level power consumption at different operating frequencies, 3.1GHz, 2.9GHz, and 2.7GHz while stacked bars show power breakdown just for the 3.1GHz run. We first use the 3.1GHz run as the example for the following analysis. For the data points of $T_{inlet\ air} \leq 33°C$ shown in Figure 23, the power elevation, mostly due to the fan power, follows the trend of the fan speed increase shown in Figure 25. As a result, the core temperature remains unchanged around $71°C$ as shown in Figure 24. This observation is different from prior study which assumed the fan power is constant [58]. Such negligence could dramatically affect the effectiveness of energy-saving strategies. Once the fan speed reaches the maximum (3100 rpm), the core temperature starts to rise and the upward trend of the power in Figure 23 also slows down. The slight power increase in this region ($T_{inlet\ air} > 33°C$) is likely due to increased leakage current caused by higher core temperature.

We now compare the results of running at different frequencies. First, when running at

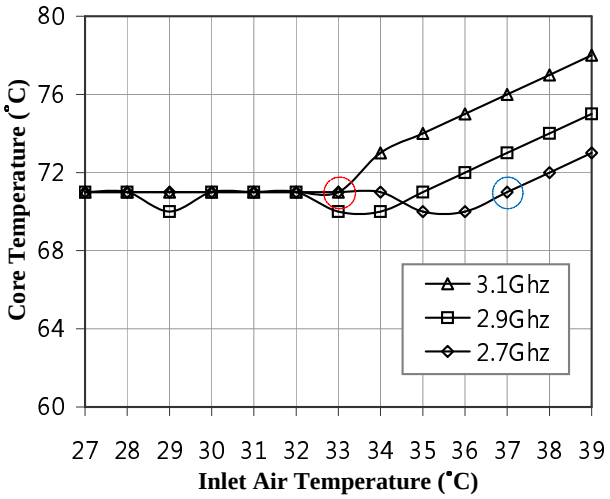


Figure 24: Inlet-air temperature versus core temperature.

lower frequencies (*e.g.*, , 2.7GHz and 2.9GHz), the system does not attempt to cool down the core temperature as shown in Figure 24. Instead, it lowers the fan speed (Figure 25) in order to reduce the fan power consumption. As shown in Figure 23, the system can save 13.5W and 18W for 2.9GHz and 2.7GHz, respectively, from the power rating of 232W for 3.1GHz at $33°C$. This power saving can be explained by the fundamental law of cooling. According to Newton's law of cooling, the rate of heat loss is proportional to the temperature difference between the object and its surroundings.

We now verify the measured power saving numbers in Figure 23 quantitatively. To eliminate the effect of the fan power and the difference of core temperatures, we pick the data points of two systems when the fan reaches its maximum speed with the same core temperature. As indicated by the circles in Figure 23, the runs of 3.1GHz and 2.7GHz reach that state when $T_{inlet\ air} = 33°C$ and $T_{inlet\ air} = 37°C$, respectively. According to Figure 24, both scenarios have the core temperature at $71°C$. Then the temperature differences between the core and its surroundings (*i.e.*, $T_{core} - T_{inlet\ air}$) are $38°C(= 71 - 33)$ and $34°C(= 71 - 37)$ for the 3.1GHz and 2.7GHz core. The 3.1GHz core has an advertised Thermal Design Power (TDP) of 80W, in other words, the cooling system, rotating the fan at the maximum speed, can remove heat generated by an 80W core when the delta

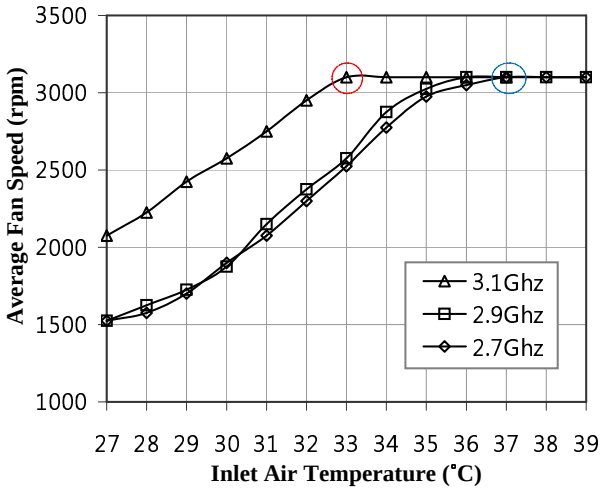


Figure 25: Inlet-air temperature versus fan speed.

temperature is $38°C$. Based on the law of cooling, the 2.7GHz system will remove heat generated by a 71.6W ($= 80W \times \frac{34°C}{38°C}$) core. Our measurement result of these two systems in Figure 23 (*i.e.*, the power difference between two dashed circles) shows a 9W difference, which closely conforms to the theoretical deduction of 8.4W.

By using the relation discussed above, we now present a simple example with respect to how to keep the core temperature constant under control while the inlet temperature goes above emergency temperature ($T_{emergency}$). Initially, we assume a server whose temperature difference between the core ($T_{core} = 70°C$) and the ambience ($T_{inlet\ air} = 30°C$) is $40°C$ when the inlet temperature is $30°C$. Now we tune down the cool air supply from the CRAC unit and subsequently the server senses the $T_{inlet\ air}$ raised to $35°C$, which is $5°C$ above $T_{emergency}$. In other words, the temperature difference (ΔT) between the core and the ambience is reduced to $35°C$. According to our previous discussion, due to the fan has reached its maximum rotation speed, the server will have to increase its core temperature by $5°C$ to $75°C$ to achieve the equilibrium, which is undesirable due to reliability issue. Another option for the server will be to reduce its own power consumption to keep the core temperature at $70°C$. Based on our prior deduction, the power draw has to be proportionally decreased to achieve this goal. Therefore, the server has to reduce its power down to $\frac{35}{40}$th of

the original power via technique, such as, DVFS to keep the core temperature from rising.

4.2.2.2 Thermal Impact on Fan Power

To build a link from $T_{inlet\ air}$ to fan power, we adopt a similar approach as in prior literature [56, 59]. First, we use the *Fan Affinity Laws* that indicates - (1) the fan power is in a cubic growth of the rotational speed; (2) the volume capacity (the amount of air) of a fan is proportional to the rotational speed. Thus, the following relations hold.

$$\begin{gathered} Fan\ Power \propto (RPM)^3 \\ Volume \propto RPM \\ Fan\ Power \propto (Volume)^3 \end{gathered} \tag{23}$$

Second, we use the *Laws of Convective Heat Transfer* that indicates that heat transfer or power (in watts) is proportional to (1) the volume capacity of air[3] and (2) the temperature difference between T_{core} and $T_{inlet\ air}$, or ΔT.

$$\begin{gathered} Heat\ Removal \propto Volume \\ Heat\ Removal \propto \Delta T \end{gathered} \tag{24}$$

Therefore, when the temperature difference ($\Delta T = T_{core} - T_{inlet\ air}$) becomes half of what it was, the volume capacity has to be doubled to maintain the cooling capacity.

$$\begin{gathered} \frac{Heat\ Removal\ Per\ Volume_{before}}{Heat\ Removal\ Per\ Volume_{after}} = \frac{\Delta T_{before}}{\Delta T_{after}} = 2 \\ \text{To make } Heat\ Removal_{before} = Heat\ Removal_{after} \\ \therefore Volume_{after} = 2 \times Volume_{before} \end{gathered} \tag{25}$$

Since the volume capacity of a fan is proportional to rotational speed, a halved ΔT will result in doubling the rotation speed. The fan now rotates twice as fast and consumes 8x more power.

$$\begin{gathered} \frac{Volume_{after}}{Volume_{before}} = \frac{RPM_{after}}{RPM_{before}} = 2 \\ \therefore \frac{Fan\ Power_{after}}{Fan\ Power_{before}} = (\frac{RPM_{after}}{RPM_{before}})^3 = 8 \end{gathered} \tag{26}$$

[3]For simplicity, we assume that the density of air is constant at the temperature range of interest throughout the dissertation.

In summary, a higher $T_{inlet\ air}$ results in a smaller ΔT and increases the fan power.

4.2.3 Details of ATAC Algorithm

In this section, we propose ATAC (Ambient Temperature Aware Capping), a system-level technique to guarantee the reliability of operations when we tune down the cooling units for improving energy efficiency. Our proposed scheme enables the inlet air supply to furnish less cooling air for saving cooling energy, and at the same time, applies ATAC to allow each server to dynamically scale down its frequency and voltage (*i.e.*, capping the performance). Local to each server, the ATAC mechanism collects various information including core temperature, inlet-air temperature, fans' rotational speed, and CPU's thermal design power (TDP), and checks if the inlet-air temperature ($T_{inlet\ air}$) is above the emergency temperature ($T_{emergency}$) to make a decision for performance capping.

Initially, the system administrator starts to operate the datacenter in a way that all the CRAC units supply cool air at the lowest possible temperature. Then the CRAC controller increases the air supply temperature from the CRAC units, which in turn will reduce the energy consumed by them [9, 49]. CRAC's discharge temperature keeps raising until the highest inlet-air temperature of a server reaches a triggering temperature point, $T_{trigger}$. At the moment that any of the servers experiences $T_{trigger}$, CRAC units now start to lower the supply air temperature. In this scenario, ATAC constantly monitors the inlet-air temperatures from each server, obtained using the thermal sensor embedded in the servers. If $T_{inlet\ air}$ stays below $T_{emergency}$, the triggering event does not occur. Otherwise, ATAC of the violating server will cap its own performance by scaling down its frequency/voltage to reduce the power consumption. Note that ATAC has to assure that the power is proportionally reduced with the delta temperature ($\Delta T = T_{core} - T_{inlet\ air}$) based on the discussion in Section 4.2.2.1.

Now we discuss the relationship between power and performance for designing an effective performance capping mechanism. In general, the performance is not proportionally reduced by the reduction of power. For instance, in the experiment shown in

Figure 23 through Figure 25, we find that when the frequency is lowered from 3.1GHz to 2.7GHz (87.1%), the power is reduced from 80W to 72W (90.0%); however, performance results from LINPACK benchmarks is only reduced from 15.3 Gflops to 15.1 Gflops (98.6%). In the previous example, if we define 90.0% as the power ratio and 98.6% as the performance ratio, then the relationship between the two is close to (Power ratio) = (Performance ratio)^7, which indicates only slight performance degradation by reduced power. To obtain a more conservative evaluation, we adopt a general power and performance model from other studies [60, 61] in which the power ratio is equal to the square of the performance ratio ((Power ratio) = (Performance ratio)^2). Based on this model, if the performance of a core degrades by 90%, its power consumption will be reduced to 81% (= $(0.90)^2$). For the following evaluation of ATAC, we use this conservative assumption.

Although ATAC is designed to exploit the inequality or non-uniformity of the inlet-air temperature among the servers in a rack, we also argue that the ATAC benefits from the uneven cooling effectiveness among servers in a datacenter. Depending on the proximity of the racks to the CRAC unit, the inlet temperature of some servers changes more rapidly than the others. When some servers take longer time to cool down, the CRAC unit cannot raise supply air temperature instantly even though all the servers are running below $T_{trigger}$. This is because of the uncertainty of future workload. If the datacenter has no information about the future workload, the CRAC unit cannot aggressively raise the room temperature but has to maintain a safety margin. In other words, to maintain $max(T_{inlet\ air})$ strictly under $T_{emergency}$, $T_{trigger}$ cannot be as high as $T_{emergency}$. Section 3.3 identified that such phenomenon is caused by non-uniform distances from CRAC units to servers. In contrast, with ATAC support for all the servers in the datacenter, the CRAC unit can increase the supply air temperature more aggressively with more relaxed safety margin as the built-in dynamic performance capping scheme in each server can respond and resiliently control the core temperature.

4.2.4 Simulation Setup

4.2.4.1 The Simulation Platform and Inputs

In this chapter, we use SimWare [56] as an evaluating platform. SimWare is the only publicly available datacenter simulator that implements a variety of critical components in a warehouse-scale computer including:

- A detailed power model of servers by utilization level and inlet-air temperature
- CRAC power models [9] by supply air temperature
- The effect of heat recirculation [41]
- The effect of the timing delay of cool air delivery from CRAC to the front plate of servers

At the end of the simulation, SimWare outputs utilization level-, power-, and latency-related statistics.

To model different datacenter settings, SimWare supports a variety of *configurable* parameters, including the number of server chassis in a simulated datacenter, the number of servers per chassis, architectural specifications for CPU and fans, task scheduling algorithms, and CRAC algorithms to control air supply. First of all, SimWare provides two different CRAC controlling algorithms; constant and dynamic. In the constant controlling algorithm, the cool air temperature supplied by CRAC does not vary (*i.e.*, constant temperature). Since CRAC does not change the supply air temperature, datacenter administrators assume the worst case scenario where all the servers are fully loaded. Since this worst case scenario is extremely rare, the constant CRAC control overcools the datacenter most of the time and thus scores low power efficiency.

The dynamic algorithm, on the other hand, changes the supply air temperature while sensing the inlet-air temperature of the servers. Algorithm 1 shows the dynamic CRAC control implemented in SimWare. CRAC starts to supply cool air at the lowest possible temperature, and raises the temperature until any server's inlet-air temperature hits a triggering temperature, $T_{trigger}$. Upon such an event, CRAC begins to lower the supply air

temperature to cool down the room temperature. In general, the inlet-air temperature of a server is computed as follows [62]:

$$T_{inlet\ air} = T_{supply\ air} + T_{recirculated\ heat} \tag{27}$$

Here, $T_{recirculated\ heat}$ represents the thermal impact caused by heat recirculation of the other servers. Note that this heat-recirculation effect is the primary reason why $T_{inlet\ air}$ varies by the location of the servers. Also, the goal of the dynamic CRAC control can be expressed as follows:

$$\forall T_{inlet\ air} < T_{trigger} \tag{28}$$

Throughout this chapter, we assume that the simulated datacenter uses the dynamic CRAC control, which dynamically changes the discharge air temperature. Therefore, we do not specifically show which $T_{supply\ air}$ is used, but show which $T_{trigger}$ is used.

Algorithm 1 Dynamic CRAC Control

Require: $T_{supply\ air} \leftarrow$ *lowest possible temperature*

loop

while $\forall$ *Server's inlet air temperature* $< T_{trigger}$ **do**

CRAC raises $T_{supply\ air}$ *for* $0.01°C/sec$

end while

while $\exists$ *Server's inlet air temperature* $\geq T_{trigger}$ **do**

CRAC lowers $T_{supply\ air}$ *for* $0.01°C/sec$

end while

end loop

Secondly, we use Google Cluster Data (GCD) [63, 64] as the input to SimWare. Google released GCD, one of the most detailed utilization traces, to public in 2011. GCD comprises 178GB of text files containing detailed information that is collected from the jobs submitted to one of the company's datacenters. The overall computing cluster has about

12500 heterogeneous computing nodes in 10 different groups [65]. Although the different groups have disparate hardware specifications, we regroup the nodes into three groups based on the CPU performance metric. This is because current SimWare only models server's power consumption by CPU utilization, but not by memory or disk utilization. In terms of normalized CPU performance, servers in GCD have three different types: 0.25, 0.5 and 1. On the contrary, in our simulated datacenter, servers are homogeneous (*i.e.*, all the servers share the same computing capacity). Since more than 92% of the servers in GCD have a normalized CPU scale of 0.5, we assume that a CPU scale of 0.5 matches to one core in the simulated datacenter. For the servers with a CPU scale of 0.25 or 1, we assume linearly decreased or increased execution time, respectively. For example, one second in a machine with a CPU scale of 0.25 corresponds to a half second in a machine with a CPU scale of 0.5. The rest of the configurable parameters for SimWare are discussed in the next section.

4.2.4.2 Specifications for Blade Servers

In our simulation, we use the same 50 by 50 heat recirculation matrix as in the original SimWare in Section 3.3. Because the number of rows in this square matrix represents the number of blade server chassis inside the datacenter, the simulated datacenter has 50 blade server chassis. We also configure each blade server chassis holds five blade servers to attain a total of 250 blade servers. Table 2 summarizes the specification of a blade server in our simulated datacenter. Each blade server has an Intel Phi, one of the Intel's anticipated products with 57 cores. However in our simulation, we only activate up to 51 cores from each server. This is because GCD utilizes only 12583 cores, or 50.332 cores (= 12583/250) per server. Therefore, we recalculate the power consumption as follows. Even though Intel's Phi is rated at 300W [66], we first subtract its maximum fan power and multiply by $\frac{51\text{cores}}{57\text{cores}}$ to obtain the maximum power in our simulation. We first assume that the fan attached to Intel's Phi consumes up to 21.6W. In this case, the maximum CPU power becomes $(300W - 21.6W) \times \frac{51}{57} = 249.1W \cong 250W$. Here, because Intel did not

Table 2: Specification of the simulated blade server.

Component name	*Specification*
CPU	57-core Intel Phi. TDP=300W [67]. When only 51 cores are active, TDP=250W
CPU Cooling Capacity	CPU fan removes heat generated by 250W when the fan rotates at the maximum speed
$\Delta T = T_{core} - T_{inlet\ air}$	When the fan rotates at its maximum speed and the CPU is at full load, The temperature difference between the processor die's temperature and the ambient air is $40°C$.
CPU Fan	Maximum speed = 4800 rpm; power = 21.6W.
Case Fans	Two more fans are located at the front and back of each server.
Fan Control	When $T_{core} < 70°C$ the priority of the fan control is in saving fan power. Otherwise, when $T_{core} \geq 70°C$, the priority is in lowering T_{core}. The cpu fan cannot be turned off and runs at 500 rpm when the server is idle. Case fans increase rotational speed proportional to the power consumption of the server and the inlet-air temperature.
Idle Power	The blade server consumes 250W plus corresponding fan power when idle.
Peak Power	The blade server consumes 565W in maximum.

reveal the detailed specification of the fan attached to Phi, we assume the same fan used in Nvidia's GTX 480 because GTX 480 had the same TDP of 250W. In summary, Intel's Phi consumes 250W when 51 cores are activated.

We also elaborate more on the detailed specification of the fan attached to Phi. First of all, the fan consumes 21.6W in maximum and removes heat generated by 250W when the the fan rotates at the maximum speed of 4800rpm. We also assume that when the fan removes the maximum power, 250W, the minimum temperature difference (ΔT) between the die and the inlet air is $40°C$. This was generated from our experiment discussed in Figure 24 where the core is at $71°C$ and the inlet-air temperature is measured at $33°C$ when the fan rotates at full speed. For simplicity, we use $40°C$ instead of $38°C (= 71°C - 33°C)$. This number is particularly important for performing ATAC. As discussed in Section 4.2.2.1 and Section 4.2.2.2, we use the temperature difference to calculate the desired power level to be achieved by DVFS. An example with respect to how to reach the desired power level was given at the end of Section 4.2.2.1.

There are two other fans with the same specification used in the server. One is located at the front panel of the server and the second one at the back. The rotational speed of these case fans are directly proportional to the power consumption of the server and inlet-air temperature. For simplicity, the boundary condition is that the fans are rotating at 4800 rpm (maximum), when the server is fully loaded at $30°C$. In addition, we assume that the goal of fan control is to save fan power and set the die temperature lower than $70°C$ for the reliability. In terms of the peak power of the blade server, we add up the idle power, peak CPU power, and all three fan powers. We first assume that the blade server consumes 250W when idle[4]. Then the peak power becomes 250W (idle power) + 250W (peak CPU power) + 3×21.6W (three fans) = 564.8W

4.2.5 Evaluation and Analysis

4.2.5.1 The Baseline Analysis

For the legacy datacenters, typical $T_{trigger}$ value ranges from $20°C$ to $30°C$, and the average $T_{supply\ air}$ is around or even lower than $15°C$ [50, 10, 51]. However, according to Intel's projection of future datacenters, high ambient temperature (HTA) datacenters will let the servers operate above $40°C$, or even more than $50°C$ [2]. Because this chapter focuses on the power optimization for future HTA datacenters, our experimental ambient temperature ranges from $40°C$ to even higher than $50°C$. Hence, throughout the chapter, we use $T_{trigger} \geq 40°C$.

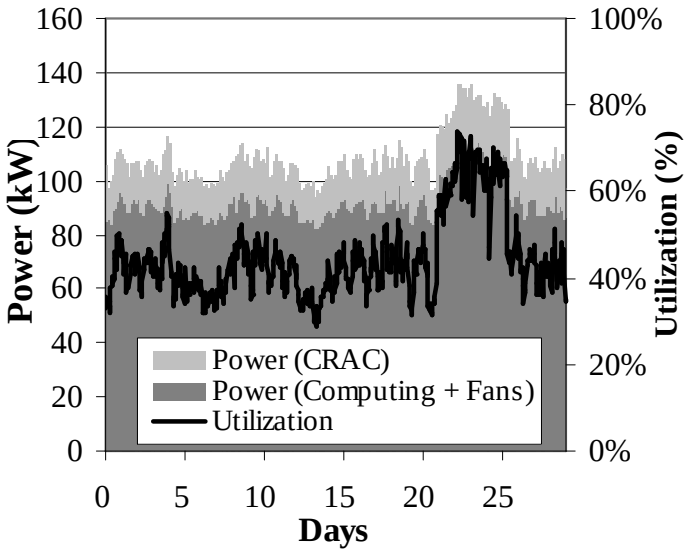
Figure 26a shows the overall utilization level of the simulated datacenter when $T_{trigger} = 40°C$. The X-axis represents the elapsed time while the primary Y-axis (left) and the background area chart show the power consumption in watts. In addition, the secondary Y-axis (right) and the solid line chart show the utilization level. As stated before, GCD contains job traces for about a month, and the average daily utilization level ranges from 40% to 60%. If we divide the time line into four consecutive weeks, the fourth week shows significantly higher utilization level. The power consumption curve for computing and cooling

[4]Typical servers consume the half of the peak power when idle [14]. When we exclude the fan power, our blade servers consume half of the peak power when idle.

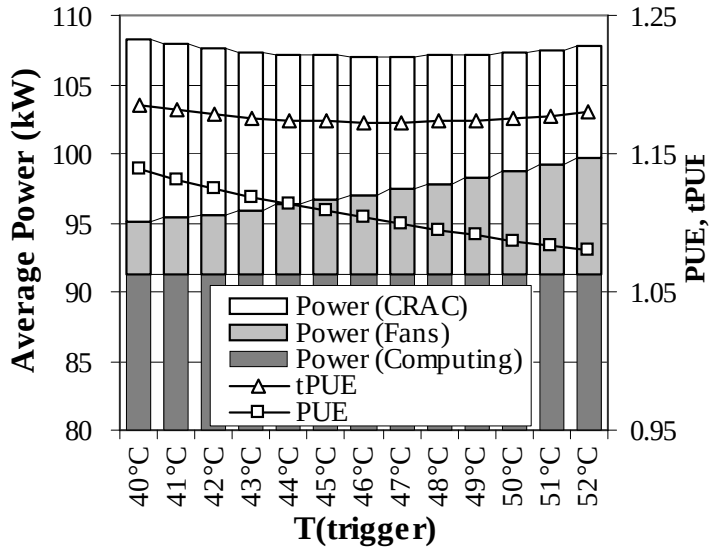
units generally tracks the utilization level. More interesting observations can be made from Figure 26b. In Figure 26b, we increase $T_{trigger}$ from $40°C$ to $52°C$ in X-axis. As we increase $T_{trigger}$, the datacenter saves cooling power while spends more on fan power. As a result, we identify that after $T_{trigger} = 47°C$, raising room temperature no more reduces the total power consumption of the datacenter. Meanwhile, PUE is monotonically decreasing as we increase the ambient temperature. This is because PUE is proportional to the ratio of the CRAC power to the servers power, if the facility power is remains constant. With reduced CRAC power and elevated server power (due to increase in fan power), PUE decreases even though the total power consumption increases. Therefore, we suggest to use tPUE for measuring the power efficiency of datacenters. Because tPUE factors out the fan power from useful computing power, we find that lower tPUE guarantees a better power efficiency.

Another important implication of higher $T_{trigger}$ is higher T_{core}. When the fan is not at the maximum rotational speed, a system can hold T_{core} even at a higher $T_{inlet\ air}$ by increasing the fan power. However, in rare situations, the following three conditions can occur at the same time. Firstly, the fans are already at the maximum. Secondly, the CPU is at the full load. Lastly, $T_{inlet\ air}$ is raised above $T_{emergency}$. In such situation, T_{core} rises to maintain the temperature difference ($= \Delta T$) between T_{core} and $T_{inlet\ air}$ constant. Figure 27 shows how rare such situations are. In Figure 27, we illustrate the distribution of T_{core} across different $T_{trigger}$ values. As we increase $T_{trigger}$ from $40°C$ to $52°C$, the maximum value of T_{core} also increases. Note that the Y-axis is in a log scale, indicating that the chances for having a higher T_{core} are rare. In other words, the T_{core} distribution has a long tail on a high temperature region. Nevertheless, if a CPU experiences $T_{core} = 90°C$ for only a few seconds in a month, the CPU must guarantee reliable operation at $T_{core} = 90°C$.

Here, we recall that high T_{core} occurs when three conditions are met at the same time. In other words, if any of three conditions can be broken, we can avoid unnecessary reliability emergencies. Our proposed ATAC breaks the second condition by sensing $T_{inlet\ air}$ and



(a) Utilization trace.



(b) Energy, PUE, and tPUE.

Figure 26: Simulated results for Google cluster data in 2011.

changes DVFS state so that the CPU cannot be fully utilized. As we show in the next section, ATAC initiates performance capping only for a small fraction of time, therefore, the overall responsiveness of the datacenter remains nearly the same while the maximum T_{core} drastically decreases.

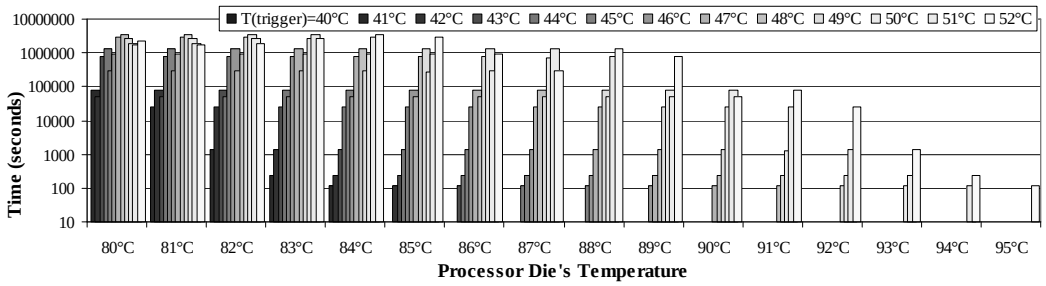


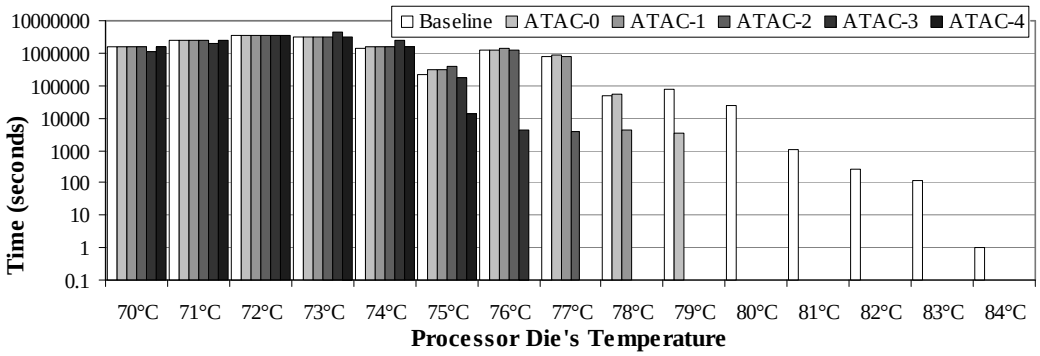
Figure 27: Distribution of core temperature when $T_{trigger}$ changes from 40°C to 52°C.

4.2.5.2 Evaluating ATAC

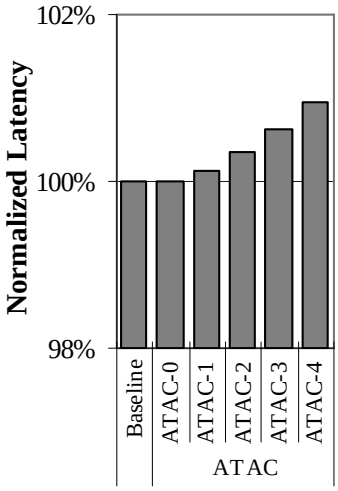
In applying ATAC mechanism to the servers, a datacenter administrator can set how aggressive will ATAC be. For example in Power Capping [21], a server with a 1000W name plate can be set to consume 900W, or even 800W by administrators' decisions. When the server's power consumption is capped at 800W, the server performs less than when it is capped at 900W. Similarly, administrators can configure the aggressiveness of ATAC. Aggressive ATAC will activate performance capping more often.

We start with the most basic strategy, ATAC-0, which activates performance capping when $T_{inlet\ air} = T_{trigger}$. In other words, $T_{emergency}$ for this configuration is $T_{trigger}$, meaning that when a server senses $T_{inlet\ air} > T_{trigger}$, the maximum performance of the server is capped. For example, we assume that $T_{trigger} = T_{emergency} = 40°C$, and one of the servers in the datacenter senses that its $T_{inlet\ air}$ is 45°C. In this case, without ATAC support, T_{core} can be as high as 85°C(= $T_{inlet\ air} + \Delta T = 45°C + 40°C$) according to the ΔT specification in Table 2. However with ATAC support, after acknowledging that $T_{inlet\ air}$ is 5°C over $T_{trigger}$, ATAC reduces the maximum power consumption of the CPU to $\frac{\Delta T - 5°C}{\Delta T}$. As a result, required temperature difference between T_{core} and $T_{inlet\ air}$ is also reduced to 35°C, and the maximum T_{core} becomes 80°C. Figure 28a shows the results of the scenario described above. In Figure 28a, the distribution of T_{core} for the baseline configuration goes as high as 84°C while ATAC-0's worst-case T_{core} is 80°C. We also define more aggressive ATAC, from ATAC-1 to ATAC-4. In ATAC-1, performance capping is activated by ATAC when

$T_{inlet\ air} = T_{trigger} - 1$, and ATAC-4 activates it when $T_{inlet\ air} = T_{trigger} - 4$. As a result, the maximum T_{core} reduces to 79°C for ATAC-1 and 76°C for ATAC-4.



(a) Core temperature for the baseline and five ATAC configurations.



(b) Normalized latency.

Figure 28: ATAC's impact on core temperature and latency when $T_{trigger} = 40°C$.

Even though ATAC-0 lowers the maximum T_{core} about 4 ~ 5°C than the baseline, the chances for activating performance capping is low. We roughly calculate how low the chances are from Figure 28a. Firstly, we add up all bars from 80°C to 84°C from the baseline. The result is 26096 seconds. Because there are about 626 million CPU seconds (= 50 chassis × 5 servers × 29 days × 24 hours × 3600 seconds) in our study, 26096 seconds is less than 0.01% of the time. In summary, ATAC-0's impact on the responsiveness of the simulated datacenter is close to 0%. As shown in Figure 28b, more aggressive ATAC such

as ATAC-4 shows the performance degradation of less than 1%.

4.2.5.3 Comparing ATAC against Power Capping and PowerNap

ATAC is unique in that it takes ambient temperature into account. Because ATAC activates performance capping from the servers at the highest inlet-air temperature, ATAC exploits temperature differences between servers. Figure 29 details the effect of ATAC on servers' performance. We first group the servers by the height in the racks. Since the simulated datacenter supplies cool air from the floor, servers near the floor has the lowest average inlet-air temperature. Therefore, servers located at the lowest to middle position do not activate performance capping for any configuration we test in Figure 29. ATAC activates performance capping only for the servers at top two positions. Even for the servers at top two positions, the performance is sacrificed only for a fraction of time when the inlet-air temperature is higher than $T_{emergency}$. Therefore, on the right-most ten bars in Figure 29, the worst case server with ATAC-3 scores 90% of the original performance. Note that Figure 29 shows the lowest performance scale of all time. In average, the performance scale of any server scores more than 99% of the original performance regardless of the location. Because ATAC exploits non-uniform inlet-air temperature among servers, ATAC outperforms the other power management schemes.

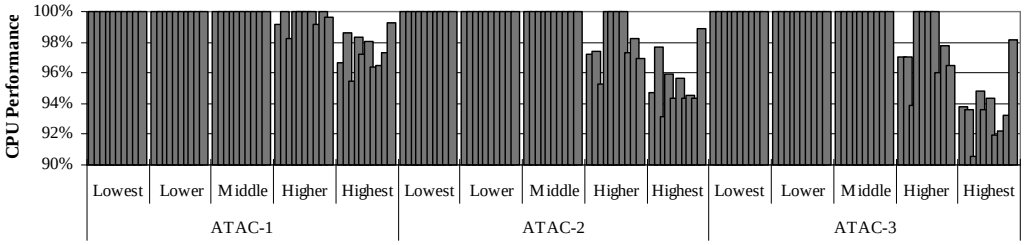
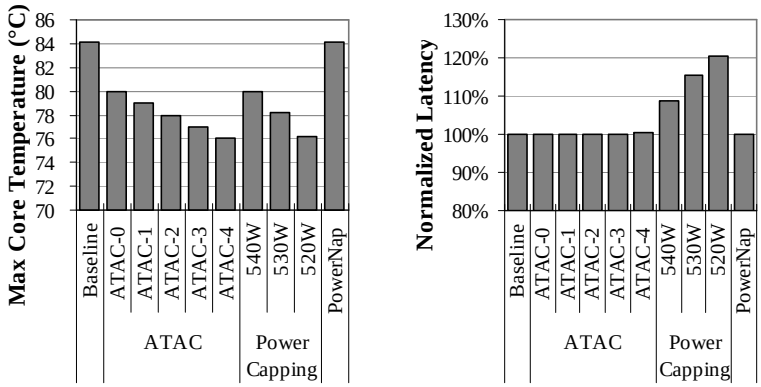


Figure 29: ATAC's impact on cpu performance (lowest value of all time) by height of servers.

Figure 30 shows the maximum T_{core} value and the normalized latency of the simulated datacenter for different power management algorithms including Power Capping [21] and

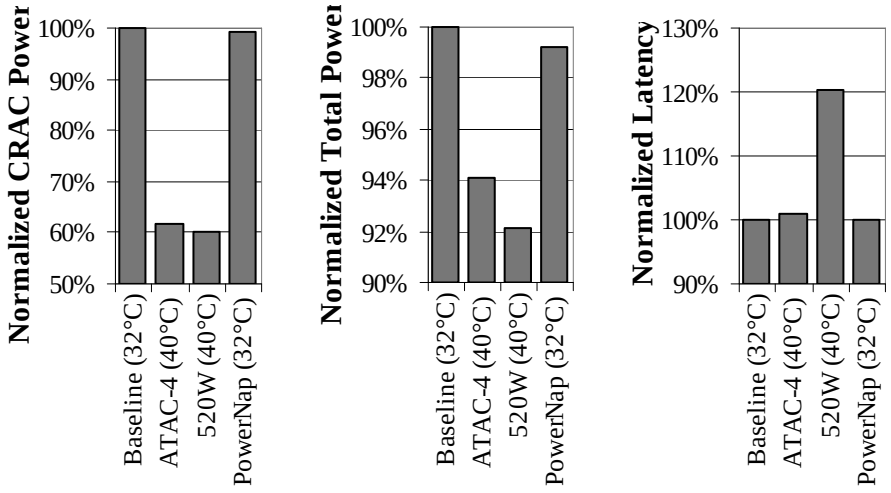
PowerNap [20]. Power Capping is a power management technique for datacenters that activates performance capping by sensing system-level power consumption and strictly limits the maximum power consumption under the bar. In our experiment, when Power Capping is available, servers' power are capped to 540W, 530W, or 520W. We also implement the ideal PowerNap. Although the original PowerNap has 300μs performance penalty for waking up from the napping state, we assume zero penalty to show the upper bound of the effectiveness of the algorithm. In addition, we use the same configurations for the baseline and ATAC-0 ~ ATAC-4 as in Figure 28.



(a) Max core temperature. (b) Normalized latency.

Figure 30: Comparing ATAC against other power management algorithms when $T_{trigger} = 40°C$.

Firstly, Figure 30a shows that ATAC and Power Capping are effective in reducing the maximum value of T_{core}. For example, when Power Capping is set to 520W, the highest T_{core} is 76°C, which is close to T_{core} of ATAC-4. However, as shown in Figure 30b, Power Capping to 520W results in 20% performance degradation while ATAC-4 shows less than 1% degradation. This is because Power Capping lowers the performance of CPU only by detecting the system-level power consumption. Even when the server burns the full power, there are no temperature emergencies when $T_{inlet\ air}$ is substantially low. Figure 30 also shows that PowerNap has no impact on T_{core} nor on the normalized latency. This is because PowerNap is not designed to control T_{core} but to save server power for achieving



(a) CRAC power. (b) Total power. (c) Normalized latency.

Figure 31: Maximum core temperature equivalent comparison.

energy proportionality [14].

4.2.5.4 Max(T_{core})-Equivalent Comparison

As discussed in Section 4.2.5.3, ATAC and Power Capping algorithms effectively lowers the upper bound of T_{core}. For example, ATAC-4, which only activates performance capping when $T_{inlet\ air}$ is higher than $T_{trigger} - 4$, lowers the maximum T_{core} value from 84.1°C to 76.0°C when it is compared to the baseline where $T_{trigger}$ = 40°C. Results from additional simulations show that the baseline datacenter without any power management mechanism must lower $T_{trigger}$ from 40°C to 32°C to achieve the same level of T_{core}. Similarly, since PowerNap has no impact on T_{core}, PowerNap also has to lower $T_{trigger}$ to 32°C for achieving the maximum T_{core} of 76°C. On the other hand, when Power Capping is available and set to 520W, the maximum value of T_{core} was the same as ATAC-4 without changing $T_{trigger}$. In summary, ATAC-4 and Power Capping set to 520W both achieve the maximum T_{core} of 76.0 ± 0.1°C while the baseline and PowerNap have to lower $T_{trigger}$ to 32°C.

We compare power consumption of all four configurations in Figure 31a and Figure 31b. The labels on X-axis show the name of four configurations and corresponding $T_{trigger}$ value in the parenthesis. Note that all configurations have the same peak T_{core} values

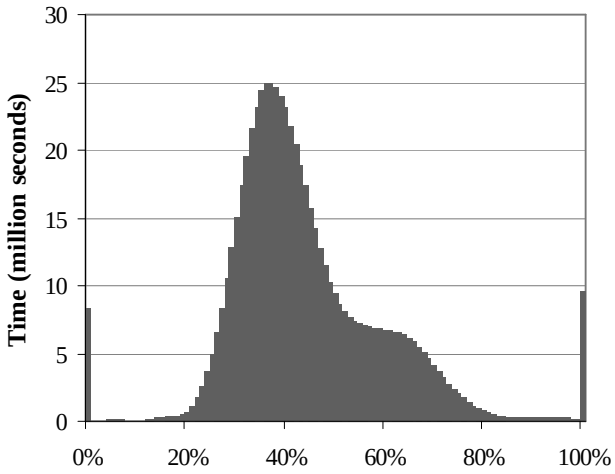


Figure 32: Per-server utilization distribution.

of 76.0 ± 0.1°C. In terms of the cooling power, savings for ATAC-4, Power Capping, and PowerNap are 38%, 40%, and 1% respectively. Such savings are translated to about 6%, 8%, and 1% savings in terms of the total datacenter power, including all the components such as computing power, fan power, and cooling power. Power Capping to 520W is the most effective power saving technique; however, it comes with the significant performance penalty. Figure 31c shows the responsiveness of the simulated datacenter. The datacenter with Power Capping set to 520W shows over 20% latency penalty. In contrast, ATAC-4's impact on the performance is negligible, less than 1%. Even though our implementation assumes the ideal PowerNap, Figure 31 shows that PowerNap has limited impact on the overall power consumption of the datacenter. The reason for such observation can be explained by Figure 32. The figure shows the distribution of the server-level utilization of the baseline configuration ($T_{trigger}$ = 40°C without any power management scheme) in seconds. As shown, servers spend most of the time in the utilization level of 20% to 80%. Servers in GCD are completely idle only for 1.3% of the time. Because PowerNap puts servers in napping state when they are completely idle, PowerNap has less than 1.3% of the headroom for this specific utilization trace. However, we also find that PowerNap can be used in conjunction with ATAC to save additional 1% of the total power consumption.

4.2.6 Related Work

Researchers have investigated increasing the supply air temperature without compromising reliability. Moore *et al.* [9] proposed a new job scheduling policy to minimize the heat recirculation effect, and Banerjee *et al.* [57] further improved it. A prior study found that when $T_{inlet\ air}$ increases, the processor cores contribute to the majority of additional power consumption [58]. Atwood *et al.* [68], however, showed that the failure rates of servers have little correlations to temperature, dust, and humidity. These studies motivated us to design system-level support that exploits the cooling inequality among the servers in datacenters.

In this work, we primarily focus on the power consumption of cooling units and servers; nonetheless, other sources of inefficiency were explored in prior research. For example, Wang *et al.* [69] and Pelley *et al.* [16] proposed efficient power delivery and smarter cluster-level power controller, and Li *et al.* [70] proposed power-efficient execution of programs. In addition, Haque *et al.* [71] proposed a new definition of service-level agreements, Green SLAs, for the clients who care about using green energy. Alleviating the peak power consumption is an important issue for datacenters [72] because their electricity bills are based on (1) the amount of energy they use and (2) the peak power that they demand. Use of fresh-air cooling [73] or renewable energy [74, 75, 76] also improves cooling efficiency of datacenters. Although ATAC achieves the same goal (*i.e.*, improving the cooling efficiency), it can be used in parallel with aforementioned techniques. For example, with ATAC support, a datacenter with free-cooling systems can exploit high temperature variations among server locations.

Similar to ATAC, Zephyr [77] discussed blade chassis-level power optimizations including fan and server power, while our study focuses on datacenter-level power optimizations including cooling power. In addition, the novelty of ATAC lies in exploiting location-dependent and regional cooling characteristics inside datacenters.

Advancements of micro-architectures and memory technologies can lead to significant

energy savings in datacenters. For example, Razor [25] allows microprocessors to operate at a lower voltage by comparing results from multiple flip-flops operating at different speeds. Razor is in fact conceptually similar to ATAC: Razor lowers a supply voltage and exploits voltage safety margins of microprocessors, while ATAC lowers cooling power and exploits temperature safety margins of datacenters. Emerging memory technologies, such as die-stacked memory [78], would also play a key role in alleviating power concerns in datacenters. Stacked DRAM caches already become practical to be deployed in large-scale servers by alleviating hardware overhead [79] and resiliency concerns [80]. These advancements could greatly reduce computing and memory power in datacenters.

CHAPTER 5

MICRO-ARCHITECTURE-LEVEL OPTIMIZATION

5.1 Micro-architecture-level Power Breakdown

As a CPU is one of the most power hungry components in a system, it is imperative to optimize power and energy consumption of CPUs [81]. In this section, we examine and understand the power distribution within the CPU. Not only can power reduction in each CPU collectively reduce the overall power consumption of all computing nodes, it also cuts the cost of thermal management hardware, such as the sizes of the heat sinks and cooling fans and the center-level cooling strategy. As a part of this effort, we will cover the power breakdown of a CPU in two different aspects. First, the power breakdown by functional modules such as the register file, fetch logic, or ALU will be included. Second, we will further analyze the power breakdown of a CPU based on different types such as active dynamic power, sub-threshold conduction, and gate leakage.

5.1.1 Per-CPU Power Breakdown by Modules

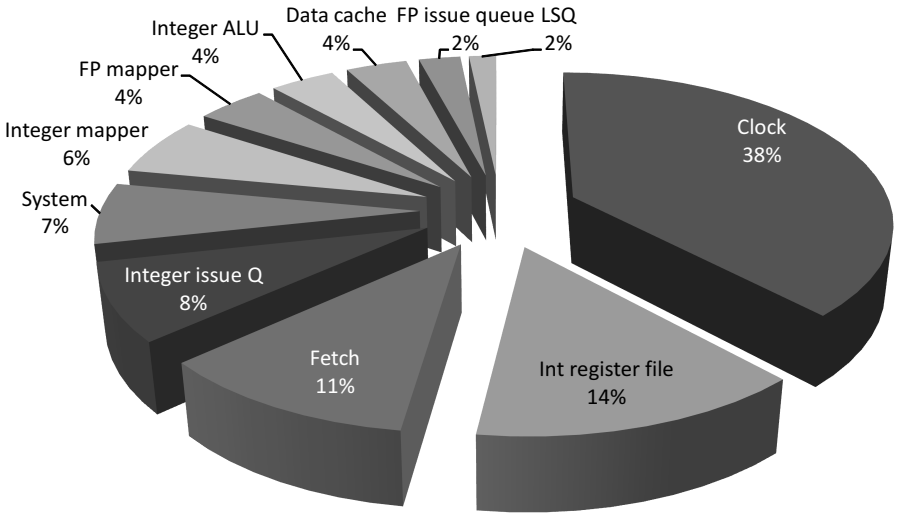


Figure 33: Power breakdown of Alpha 21264 [82]

Although there is a scarcity of public literature that breaks down the power distribution of a modern out-of-order (OoO) microprocessor, there were some attempts from both academia and industry that analyzed, modeled, and simulated the power consumption of sophisticated processors at the micro-architectural level. Figure 33 and Figure 34, both based on the DEC Alpha processor, detail and illustrate such power distribution. Figure 33 shows the power breakdown of an Alpha 21264 processor running *gzip* at 600MHz. These numbers were generated using the micro-architectural Wattch power model integrated with the cycle-level Alpha-sim simulator [82]. Given that the Alpha 21264 processor is a four-wide superscalar microprocessor with OoO execution, speculative execution, and large instruction queues for both integer and floating-point instructions, the power breakdown obtained by modeling this microprocessor will be a good representative for today's high-performance processors. From Figure 33, one can easily find that the clock tree actually accounts for more than one third of the total power dissipation. Note that, the clock signal itself is the fastest switching part of the entire chip, and this has to be done regardless of the modular utilization in the CPU. For example, the clock signal would change the logical state of the floating-point functional unit every cycle even if only an integer application is being executed. Such unnecessary power waste can be eliminated if more advanced circuit techniques such as unit-level, fine-grained clock-gating or dynamic voltage frequency scaling (DVFS) are applied. We will discuss more of these techniques in subsequent sections. To elaborate more about the clock distribution, it is worth mentioning that the Alpha 21264 processor uses a metal grid that covers the entire die area for distributing the clock signal. A metal grid for clock distribution is known to be the most effective (but not necessarily the most efficient) way of distributing clock signal with minimum clock skew to all the parts of the chip [83]. As a result, this lets a CPU run at a higher operating frequency than other types of clock distribution network such as H-tree for IBM S390 or length-matched serpentine structure for Intel P6. However, this clock distribution network, a metal grid, has a main drawback that it consumes more power than other alternatives due to its large

capacitance. Next to the clock signal, the integer register file accounts for 14% of the total power. Because these numbers are generated by running *gzip*, an integer application, the integer register file is heavily used. The accumulated OoO logic accounts for 20% of the total power consumption: 8% for the integer issue queue, 6% for the integer mapper (for register renaming in integer registers), 2% for the floating-point issue queue, and 4% for the floating-point mapper. In exchange for higher performance by exploiting instruction-level parallelism, the power portion of the OoO-related logic is larger than those of the data cache (4%) and the functional units (4%).

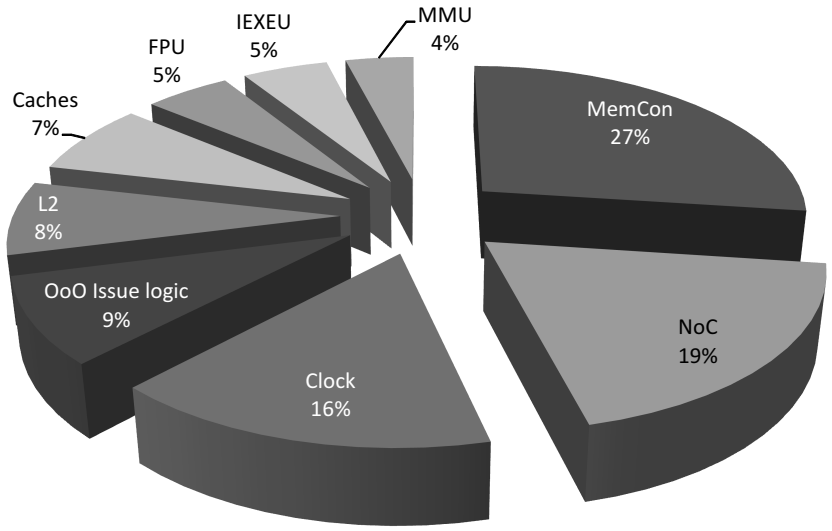


Figure 34: Power breakdown of Alpha 21364 [84]

On the other hand, Figure 34 shows the power breakdown of Alpha 21364 microprocessor generated by an integrated framework called McPAT that models power, area, and timing done by HP Labs. The Alpha 21364 processor is the successor of Alpha 21264 with minor changes on the core design with major differences on other supplementary logic including an on-die memory controller ("MemCon" in Figure 34), L2 cache, and network on chip controller ("NoC"). The design philosophy of Alpha 21364 was to improve bandwidth of the memory subsystem as well as maintaining scalability for future many-socket systems. With this objective, the memory controller and network on chip controller have

become the most power-consuming components — accounting for almost half (46%) of the entire chip power budget. For the rest of the chip, the clock distribution accounts for 16% while the OoO issue logic is about 9%.

5.1.2 Per-CPU Power Breakdown by Sources

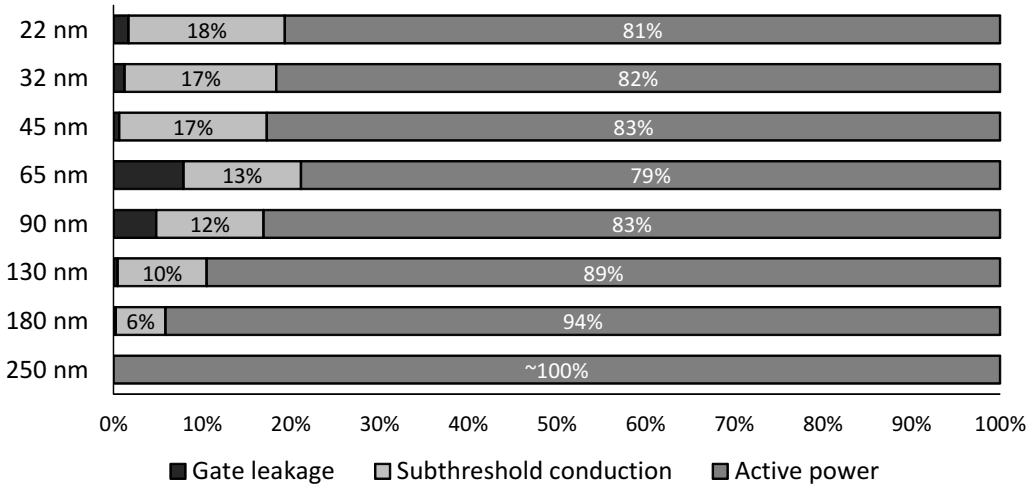


Figure 35: CMOS leak power trend by fabrication process technologies [84] [85] [86]

Figure 35 illustrates the power breakdown of a CPU by sources such as active power, sub-threshold conduction (sub-threshold leakage), or gate leakage across different fabrication process technologies. These data are collected from multiple sources [86, 84, 85]. As the feature size shrinks, as shown in Figure 35, the portion of the sub-threshold conduction continues to increase and reaches almost 20% of the total power in the 22*nm* technology node. This increasing trend is a trade-off for reducing the active power. To lower the power of a processor, designers employ lower supply voltage (V_{dd}) as the active power of a CMOS device is proportional to V_{dd}^2. When V_{dd} was high (e.g., $5V$), CMOS gates can be operated at relatively high threshold voltages (e.g., $V_{th} = 700mV$). Due to the high threshold voltage, sub-threshold leakage current were negligible as shown in the following formula where I_{off} is the sub-threshold leakage current and s is the sub-threshold swing in $mV/decade$ [87].

$$I_{off} \propto 10^{-\frac{V_{th}}{s}} \tag{29}$$

According to Equation (29), for a given sub-threshold swing, the sub-threshold leakage current is exponentially and negatively proportional to the threshold voltage. Meanwhile, V_{dd} has been lowered from $5V$ to sub-$1V$ today, V_{th} was also scaled down to $200mV$. For a sub-threshold swing of $100mV/decade$, every $100mV$ drop in V_{th} will cause ten times more sub-threshold leakage current. On the other hand, gate leakage is also exacerbated as the technology node advances. The increasing trend was because of the fact that with technology scaling, the capacitance of the gate oxide material in a MOSFET also scaled down. Equation (30) shows the relationship of capacitance (C) with the dielectric constant ($kappa$), area (A),permittivity of free space (ε_0), and insulator thickness (t).

$$C = \frac{\kappa \varepsilon_0 A}{t} \tag{30}$$

Since smaller fabrication process technology reduces area (A) of the gate oxide, the overall capacitance of the gate oxide becomes smaller, which increases the gate leakage current. As an alternative method for increasing the capacitance of the gate oxide material, material with higher κ value has been used since $45nm$ fabrication process technology, *e.g.*, Intel's high-κ metal gate technology revolution. As a result, with the "High-κ" material, the gate leakage has almost disappeared in Figure 35 since $45nm$.

5.2 Emerging Solid-state Memory Technologies

There are several emerging memory technologies looming on the horizon to compensate the physical scaling challenges of DRAM. Phase change memory (PCM) is one of such candidates proposed for being part of the main memory in computing systems. One salient feature of PCM is its multi-level cell (MLC) property which can be used to multiply the memory capacity at the cell level. However, due to the nature of PCM that the value written to the cell can drift over time, PCM is prone to a unique type of soft errors, posing a great challenge for their practical deployment. To address this reliability issue, many researchers

proposed material-based or architectural solutions. In this section, we analyze the resistance drift problem using both analytical models and Monte Carlo simulation and show the fundamental limit in prior architectural solutions. According to our findings, four-level PCM is unusable given its soft error rate and scrubbing time needed.

5.2.1 Background

Phase-change memory (PCM) is viewed as a promising alternative to dynamic random access memory (DRAM) for future computing systems. PCM stores data by changing the state of the material made of Ge, Sb, and Te (GeST). The state of PCM switches back and forth between an amorphous state and a crystalline state on microscopic level. The amorphous and crystalline states indicate high and low resistance states, respectively, which represent the value of data stored in the respective PCM cell. More specifically, a PCM cell turns into an amorphous state if the temperature of the cell is raised up to the melting point and then lowered relatively quickly. When the PCM cell is in the amorphous state, the resistance of the cell is measured around 10^6 Ohms. On the other hand, if the PCM cell is heated up to a certain temperature below the melting point and then cooled down relatively slowly, it becomes a crystalline state. When the PCM cell is in the crystalline state, the resistance is measured around 10^3 Ohms.

While adjusting the temperature and cooling time of PCM cells, researchers have learned that the resistance value of the PCM cells continuously changes from 10^3 Ohms to 10^6 Ohms. In other words, the resistance value can be found anywhere in between the crystalline state (10^3 Ohms) and the amorphous state (10^6 Ohms). Based on the understanding, multi-level cell (MLC) PCM has been studied to utilize intermediate resistance states between the crystalline and amorphous states so that the MLC PCM can store more data per cell than single-level cell (SLC) PCM.

However, MLC PCM needs more precise control over the resistance range of the cells than SLC PCM. To do so, the MLC PCM requires an iterative-writing mechanism that reads the resistance value of a cell immediately after the cell is written so that the mechanism is

Table 3: Configuration Variables of Four-level Cell PCM When t_0 = 1 s.

Storage Level	Data	$\log_{10} R$		α	
		μ_R	σ_R	μ_α	σ_α
0	01	3.0	$\frac{1}{6}$	0.001	$0.4 \times \mu_\alpha$
1	11	4.0		0.02	
2	10	5.0		0.06	
3	00	6.0		0.10	

able to confirm whether the cell is correctly written and determine whether a rewriting operation is necessary. As a result, the iterative-writing mechanism adversely affects the write latency of the MLC PCM. Recent studies show that a four-level PCM is approximately 4x ~ 8x slower than SLC PCM in terms of write latency [88].

In addition, MLC PCM has to deal with reliability challenges arising from the fact that the resistance level of cells tends to drift or rising over time and leading to soft errors. Though this problem is more evident in MLC PCM than in SLC PCM, scientists have focused on developing MLC PCM because it significantly increases the total capacity.

In light of those problems, we introduce mathematical error model that is used to calculate soft error rates of MLC PCM for the first time. With the mathematical model, we evaluate existing error-reducing techniques including memory scrubbing and error-correcting codes. Based on the evaluation, we show that four-level cell (4LC) PCM, the most conservative form of MLC PCM, is not a suitable alternative to DRAM as main memory because of its high soft-error rates.

5.2.2 Mathematical Soft Error Model and Validation

On the basis of a power-law model, Ielmini *et al.* [89, 90] reduced the resistance drift of PCM into as

$$R_{drift}(t) = R \times \{\frac{t}{t_0}\}^{\alpha}, \tag{31}$$

where R and t_0 are normalization constants and α is a drift exponent. To obtain Equation (31), Ielmini *et al.* [89, 90] conducted iterative experiments to measure the resistance

drift of reset and set states of PCM. Through the iterative experiments, the drift exponent of the reset state was found substantially larger than that of the set state. The finding indicates that the drift exponent increases directly in proportion to the portion of the amorphous state in a PCM cell.

We are aware of the fact that the resistance level of cells tends to drift, rising over time and leading to soft errors in MLC PCM. In other words, resistance drift makes MLC PCM unreliable. To estimate reliability impact of the resistance drift, we first deals with the normalization constants R and t_0 and the drift exponent α, referring to Nirschl *et al.* [91].

In Nirschl *et al.* [91], iterative-writing mechanism is performed to adjust programed resistance R_p into a certain resistance range. In such a case, $\log_{10} R_p$ is shown to follow a normal Gaussian distribution. Based on their study, we make an assumption that a *log* of R, or $\log R$, from Equation (31) follows a normal distribution $N(\mu_R, \sigma_R^2)$. Nirschl *et al.* [91] also stated that for a given state, a programmed resistance should fall within the range of $10^{\mu_R \pm 2.75\sigma_R}$ Ω, and upper and lower sensing boundaries should fall within the range of $10^{\mu_R \pm 3.00\sigma_R}$ Ω. Based on that, we assume the drift exponent α of Equation (31) follows a normal distribution of $N(\mu_\alpha, \sigma_\alpha^2)$. We use the values of the parameters indicated in the previous studies [92, 93], and Table 3 summarizes our analysis.

MLC PCM causes a soft error when the resistance level of its cell drifts and rises above the upper boundary of its programmed state. Using the upper and lower sensing boundary values presented above, we find out that the soft error occurs when the condition represented below is met.

$$R_{drift}(t) > 10^{\mu_R + 3\sigma_R}. \tag{32}$$

Equation (32) and Table 3 show that the target resistance values are 10^3, 10^4, 10^5, and $10^6\Omega$ for the four storage levels, and the three sensing boundary values are between two adjacent storage levels, $10^{3.5}$, $10^{4.5}$, and $10^{5.5}\Omega$. From these numbers, we learn that a soft error occurs when the resistance value of a PCM cell for storage level 2 is identified larger than $10^{5.5}$ Ω. In such a case, the PCM cell is identified as storing a resistance value for the

upper storage level, storage level 3.

Now we can obtain the probability of the soft error. First, we assume that $\log_{10} R$ and α follow normal distributions as described in Table 3. Then we define that m equals to $\log_{10} R$, and n equals to $\log_{10} t$. In turn, we reduce Equation (31) into the following Equation (33) using m and n.

$$\log_{10}(R_{drift}(t)) = \log_{10} R + \alpha \log_{10} t = m + n\alpha. \tag{33}$$

With Equation (32) and Equation (33), we can rewrite the condition that the soft error generates as

$$m + n\alpha > \mu_R + 3.00\sigma_R$$

$$n\alpha > \mu_R + 3.00\sigma_R - m,$$

where $n\alpha$ follows a normal distribution $N(n\mu_\alpha, (n\sigma_\alpha)^2)$ because α follows a normal distribution $N(\mu_\alpha, \sigma_\alpha^2)$. The probability that $n\alpha$ is larger than $\mu_R + 3\sigma_R - m$ is calculated as

$$\begin{aligned} &\text{(Probability of soft error for a given } m) = 1 - \Phi(\frac{\mu_R + 3\sigma_R - m - n\mu_\alpha}{n\sigma_\alpha}) \\ &\text{where } \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-x^2/2} dx. \end{aligned} \tag{34}$$

In turn, we obtain the probability density function of a random variable m, $f(m)$ of Equation (35), using the iterative-writing mechanism that repeats a write-and-verify sequence until $\log_{10} R$ falls into the range between $\mu_R + 2.75\sigma_R$ and $\mu_R - 2.75\sigma_R$.

$$\begin{aligned} f(m) &= \begin{cases} \frac{1}{K}\phi(\frac{m-\mu_R}{\sigma_R}) & \mu_R - 2.75\sigma_R < m < \mu_R + 2.75\sigma_R \\ 0 & \text{otherwise,} \end{cases} \\ &\text{where } K = \int_{\mu_R+2.75\sigma_R}^{\mu_R-2.75\sigma_R} \phi(\frac{m-\mu_R}{\sigma_R}) dm, \\ &\text{and } \phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}. \end{aligned} \tag{35}$$

Knowing that a random variable m has a certain range, $\mu_R - 2.75\sigma_R < m < \mu_R + 2.75\sigma_R$, we reduce Equation (34) into the following probability function in a time domain ($t = 10^n$).

$$\begin{aligned} &\text{(Probability of soft error)} \\ &= \int_{\mu_R+2.75\sigma_R}^{\mu_R-2.75\sigma_R} (1 - \Phi(\frac{\mu_R + 3\sigma_R - m - n\mu_\alpha}{n\sigma_\alpha})) f(m) dm \end{aligned} \tag{36}$$

The equations presented above, including Equation (36), are verified using an independent Monte Carlo simulator. We implement the simulator in accordance with the following operating steps: (1) random number generator, 2) main loop, 3) $R_{drift}(t)$ calculator, 4) $R_{drift}(t)$ evaluator and 5) repeater. In the first step, the random number generator generates random numbers from a Gaussian distribution at a given mean and variance. The second step picks corresponding R and α from Table 3, and then the simulator falls into the main loop. The simulator repeats picking R and α until $\mu_R - 2.75\sigma_R \leq log_{10}R \leq \mu_R + 2.75\sigma_R$ for the purpose of emulating the iterative writing mechanism. Once R and α in desired ranges are picked, the simulator turns into the third step that calculates $R_{drift}(t)$ using Equation (31). In the fourth step, the simulator determines a soft error occurs if $\log_{10} R_{drift}(t)$ is larger than $\mu_R + 3.00\sigma_R$. Lastly, the simulator repeats the main loop one billion times and counts the number of soft errors to obtain the soft error rate. For example, in the case that ten soft errors are generated out of one billion trials, the soft error rate is amount to 10^{-8}.

The simulation results are shown in Figure 36 and Table 4. Here, soft error rates for set state (storage level 0) and reset state (storage level 3) are not shown because a soft error does not occur in storage level 3 even if the resistance drifts, and the soft error rate of storage level 0 is negligibly low. Specifically, Mathematica 8.0 shows that the error rate of storage level 0 first turns into a non-zero value, 2.3×10^{-18}, at $t = 2^{35}$ (1090 years). Likewise, three data points for storage levels 1 and 2 are omitted and marked as "too small" because the simulator could not find error after running the main loop one billion trials or Mathematica 8.0 is not able to evaluate Equation (36). Comparing Equation (36) to the results of the Monte Carlo simulation obtained independently from Equation (36), we prove the validity of Equation (36).

One salient observation made from this experiment is that researchers need analytical models in studying soft error rates of a new technology. The Monte Carlo simulation could not identify soft errors lower than 10^{-8} from a billion trials, which is already orders of magnitude higher error rates than that of DRAM. In other words, to detect errors from

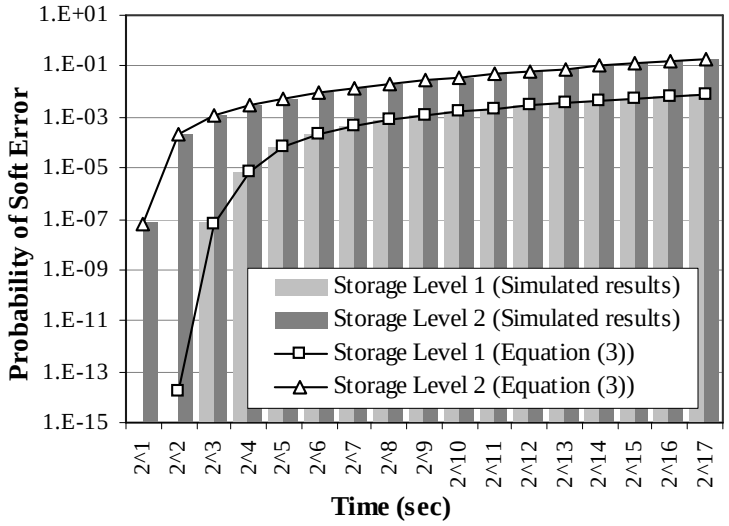


Figure 36: Probability of Soft Error of Four-level Cell PCM Over Time

the odd of 10^{-11}, Monte Carlo simulation must test several trillion trials, and this can take months and years to finish. One of major contributions of this work is that we propose a closed-form expression of soft-error rates of MLC PCM as shown in Equation (36).

5.2.3 Evaluating Four-level Cell PCM in Light of Reliability

It is obvious from Table 4 that 4LC-PCM is not suitable as a main memory because of high error rates. Various studies have been proceeded to alleviate soft errors and build drift-tolerant PCM including error correction schemes [92, 94, 95, 93], data encoding schemes using relative resistance difference [95, 94], a reference cell scheme [96], a time-aware drift estimation scheme [93], and most recently an efficient scrubbing scheme [92]. Among them, we evaluate the reliability of MLC PCM based on the efficient scrubbing scheme because it is a recently introduced technique and gaining more attentions than the others lately. Specifically, we utilize the most recent study published by Awasthi *et al.* [92] for our evaluation.

Awasthi *et al.* [92] introduced a method of reducing the soft error rate using a memory scrubbing scheme and an error correction scheme. The two schemes are combined to

Table 4: Probability of Soft Error of Four-level Cell PCM

	Storage Level 1		Storage Level 2	
Elapsed Time (sec)	Equation (36)	Simulation	Equation (36)	Simulation
2	(too small)	(too small)	5.85E-06%	7.40E-06%
2^2	1.59E-12%	(too small)	0.02%	0.02%
2^3	5.85E-06%	7.40E-06%	0.12%	0.12%
2^4	7.45E-04%	7.57E-04%	0.28%	0.29%
2^5	0.01%	0.01%	0.52%	0.53%
2^6	0.02%	0.02%	0.85%	0.86%
2^7	0.05%	0.05%	1.30%	1.31%
2^8	0.08%	0.08%	1.90%	1.91%
2^9	0.12%	0.12%	2.67%	2.68%
2^{10}	0.17%	0.17%	3.64%	3.66%
2^{11}	0.22%	0.22%	4.84%	4.87%
2^{12}	0.28%	0.29%	6.29%	6.32%
2^{13}	0.35%	0.36%	7.99%	8.04%
2^{14}	0.43%	0.44%	9.95%	10.01%
2^{15}	0.52%	0.53%	12.16%	12.24%
2^{16}	0.62%	0.63%	14.61%	14.70%
2^{17}	0.73%	0.74%	17.27%	17.38%

reduce the error rate into a level suitable for main memory. Notwithstanding the most efficient scheme, we find that the soft error rate of 4LC PCM is substantially higher than that of DRAM[1].

5.2.3.1 Estimating Scrubbing Overhead

In this section, we discuss in further details about the soft error rates (SERs) of 4LC-PCM and DRAM, and show 4LC-PCM is not a feasible alternative to DRAM in light of reliability. First, we presume that a basic access unit is a 16GB PCM main memory using a 256B data block[2] as described in prior literature [98, 99]. The read and write latencies of SLC PCM are known as 120ns and 150ns, respectively, as indicated in a recent paper, Choi *et al.* [100]. That being said, we assume that MLC PCM spends at least 1μs in scrubbing one cache line because MLC PCM necessitates the iterative-writing mechanism. Lastly,

[1]Soft error rates (SER) for DRAM are reported to be from $25,000 \sim 75,000$ FIT per Mbit, or $25 \times 10^{-12} \sim 75 \times 10^{-12}$ per bit-hour [97] on average.

[2]A last-level DRAM cache with larger capacity is used to hide PCM access latencies. We assume that its cache-line size is 256B.

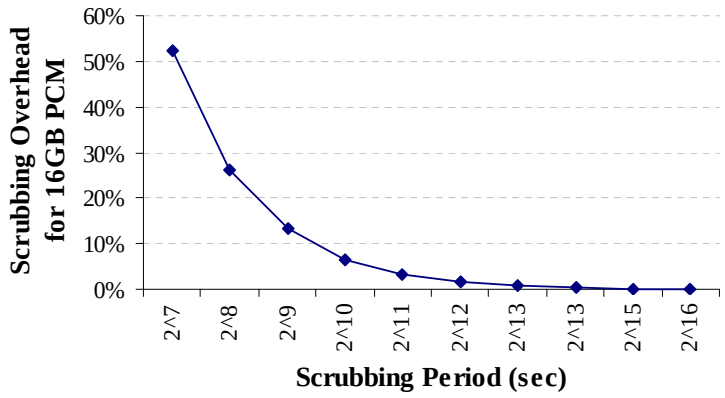


Figure 37: Scrubbing Period Versus Scrubbing Overhead

we assume that each of the storage levels occurs with the same probability.

Figure 37 illustrates scrubbing overhead in the domain of scrubbing period and the scrubbing overhead. The scrubbing overhead denotes (Time used for scrubbing)/(Scrubbing period). As the basic access unit of the 16GB PCM has 64M cache-lines, it takes 67.1 seconds ($= 64M \times 1\mu s$) to scrub the entire PCM. If the scrubbing period is set for 45 minutes as the same as in a typical DRAM memory system [97], the SER of a 4LC-PCM cell for storage level 2 comes close to 5%, still much higher than the SER of DRAM. Accordingly, we learn that 4LC-PCM does not provide reliability so much as to function as main memory in place of DRAM even with scrub mechanisms.

Table 4 shows that in the dramatic scenario that the memory controller of 4LC-PCM performs only the scrubbing operations and nothing else, the SER of storage level 2 still remains as high as 0.9%. To main SER in the rage of DRAM and still reduce the scrubbing overhead, the maximum capacity of PCM must be limited. The next section discusses the impact of reducing maximum capacity of PCM to the scrubbing overhead and SER.

5.2.3.2 Lower Soft Error Rates by Reducing Capacity

Limiting the maximum capacity of 4LC-PCM is one way to lower the SER of 4LC-PCM. Like in Section 5.2.3.1, we assume that the capacity of 4LC-PCM is 16GB in calculating the scrubbing overhead. If the capacity is assumed as 8GB, the scrubbing overhead reduces

Table 5: Maximum Capacity of Four-level Cell PCM by Soft Error Rates and Scrubbing Overhead

Scrubbing Period (sec)	$SER_{combined}$	Scrubbing Overhead		
		100.0%	12.5%	1.0%
2	1.46E-06%	488MB	61.0MB	4.88MB
2^2	0.005%	977MB	122MB	9.77MB
2^3	0.030%	1.95GB	244MB	19.5MB
2^4	0.071%	3.91GB	488MB	39.1MB
2^5	0.132%	7.81GB	977MB	78.1MB

by half as the overhead increase in proportion to the capacity. In the same sense, a lower SER is obtained if the capacity is further reduced.

We calculate the maximum available capacity of 4LC-PCM in a given SER and scrubbing overhead combination, as indicated in Table 5. The leftmost column of Table 5 shows the scrubbing periods seen by each 256B memory block, and the next column shows combined SERs representing an *average* SER of the four states of 4LC-PCM. The combined SERs are approximately one fourth of the SERs of storage level 3 because storage level 3 has a much larger SER than the other storage levels. Table 5 also shows the maximum capacity at a three different degree of scrubbing overhead. In case of 100% scrubbing overhead, the memory controller is not able to handle any service request delivered from its upper level of the memory hierarchy. Table 5 also shows 12.5% scrubbing overhead that can be considered as an upper bound as opposed to impractical 100% overhead. In addition, Table 5 presents the maximum capacity for 12.5% and 1.0% scrubbing overhead, respectively. For instance, when 4LC-PCM is set to have 1.0% scrubbing overhead and spend 99% of its time servicing memory request, the 4LC-PCM can merely have 4.88MB of maximum capacity to maintain 1.46E-06% of average SER. It has been known that scrubbing can be proceeded in parallel if 4LC-PCM has more than one bank or rank. In other words, while one bank is being scrubbed, the other bank can respond to a service request from the CPU. However, even 4LC-PCM with four ranks and four banks does not

meet the capacity required for a main memory. The maximum capacity of such 4LC-PCM is 78.1MB that is much lower than the required capacity. In sum, reducing the capacity of 4LC-PCM does not render 4LC-PCM into a feasible technology because the maximum capacity becomes too small to be used even though the SER can be lowered as a result of the reduced capacity of 4LC-PCM.

5.2.3.3 Use of Error-Correcting Codes

The SER of 4LC-PCM can be lowered using error-correcting codes (ECC). Among various ECC schemes, Hamming code error correction [101] is commonly applied to server main memory as industry standard (72,64). The error correction can be implemented simply by adding 8 redundant bits to 64 bits data[3]. Furthermore, stronger ECC, for example, BCH codes can be used to correct multiple bit errors. More specifically, the BCH codes [102, 103] correct 8, 16, 24, or 40 bit errors in 256, 512, 1024 bytes of data based on the size of the redundant bits. However, the BCH codes have disadvantage to the (72,64) Hamming code in that the BCH codes needs more computing time and power for decoding. For the reasons, the BCH codes are not frequently applied to delay sensitive devices such as main memory; however, they are more suitable for slower devices including NAND-based storage. In this section, we use (72,64) Hamming code and BCH codes together to calculate the error rates of 4LC-PCM. We refer to the combined SER as defined in the previous section and assume the data size to 256 bytes for every ECC evaluation.

(72,64) Hamming code cannot correct two or more bit errors in 72 bits data because the code only corrects one bit error. Since 36 4LC-PCM cells are necessary to store the 72 bits data, the probability of occurrence of multiple bit errors out of 36 cells is derived as follows. From Table 3, we know that changing one storage level affects one bit of two bit data at most. In accordance, two bit errors generate only when two 4LC-PCM cells are

[3]Overhead is 12.5%.

changed as a result of resistance drift.

$$\begin{aligned} &\text{Probability of having at least two bit errors} \\ =&P_{error}(64b) = 1 - P(\text{no errors}) - P(\text{one bit error}) \\ =&1 - (1 - SER_{combined})^{36} \\ &- \binom{36}{1}(1 - SER_{combined})^{35}(SER_{combined}) \end{aligned} \tag{37}$$

In turn, we calculate the probability that a uncorrectable error occurs in 256 bytes data, using the scrubbing period, scrubbing overheads, and SER obtained from Table 4. 256 bytes data comprises 32 blocks where each block has 64 bits. Accordingly, any of the 32 blocks should not cause an error to reconstruct the 256 bytes data. Therefore, the probability of experiencing uncorrectable error for 256 bytes is represented as

$$P_{error}(256B) = 1 - (1 - P_{error}(64b))^{32}, \tag{38}$$

where $P_{error}(64b)$ denotes the result of Equation (37).

Table 6 shows in the fourth column the result values of $P_{error}(256B)$ when (72,64) Hamming code is applied. From the error rates, we learn that although (72,64) Hamming code lowers the error rates, the error rates still prevent this technology from practical use. Stronger ECC is necessary to further reduce the error rates even though it leads to a large computational overhead.

We now calculate the probability that an uncorrectable error occurs in 256 bytes data when stronger ECC than (72,64) Hamming code is applied. BCH-8, BCH-16, BCH-24, and BCH-32 are examples that are stronger than (72,64) Hamming code. BCH-8 adds 12 redundant bytes and corrects up to 8 bits errors, and BCH-16 adds 24 redundant bytes and correct up to 16 bits errors[4]. We obtain the probability that *n* or more bit errors occur out

[4]Overheads are 4.7% and 9.4%.

Table 6: Probability of Uncorrectable Errors by $SER_{combined}$ for 16GB 4LC-PCM under (72,64) Hamming code

		$P_{error}(256B)$	
Scrubbing Period (Overheads)	$SER_{combined}$	No ECC	(72, 64)
2^7 sec (52.4%)	0.325%	96.4%	18.0%
2^8 sec (26.2%)	0.475%	99.2%	33.7%
2^9 sec (13.1%)	0.668%	99.9%	54.3%
2^{10} sec (6.6%)	0.91%	100%	75.1%
2^{11} sec (3.3%)	1.21%	100%	90.3%
2^{12} sec (1.6%)	1.57%	100%	97.6%

Table 7: Probability of Uncorrectable Errors by different strength of BCH codes and $SER_{combined}$ for 16GB 4LC-PCM

		$P_{error}(256B)$			
Scrubbing Period (Overheads)	$SER_{combined}$	BCH-8 (256B+12B)	BCH-16 (256B+24B)	BCH-24 (256B+36B)	BCH-32 (256B+48B)
2^7 sec (52.4%)	0.325%	0.949%	2.96E-5%	4.11E-11 %	(too small)
2^8 sec (26.2%)	0.475%	7.38%	4.00E-3%	1.09E-7%	6.24E-12%
2^9 sec (13.1%)	0.668%	29.2%	0.184%	6.68E-5%	3.65E-9%
2^{10} sec (6.6%)	0.91%	64.0%	3.08%	1.09E-2%	6.17E-6%
2^{11} sec (3.3%)	1.21%	90.0%	20.5%	0.53%	2.43E-3%
2^{12} sec (1.6%)	1.57%	98.7%	58.9%	7.83%	0.22%

of m bits data using Equation (37).

$$\begin{aligned}&\text{Probability of having at least } n \text{ bit errors out of } m \text{ bits}\\ &= 1 - \sum_{k=0}^{n-1} \binom{m}{k} (1 - SER_{combined})^{m-k} (SER_{combined})^{k}.\end{aligned} \tag{39}$$

Table 7 shows the result values of Equation (39). In case the scrubbing period is 2^7 seconds and the scrubbing overhead is 52.4%, $P_{error}(256B)$ are obtained as 0.949% and 2.96×10^{-5}% for BCH-8 and BCH-16, respectively. Note that the error rates, 0.949% and 2.96×10^{-5}%, are much smaller than the error rate, 18%, with (72,64) Hamming code. Nonetheless, the error rates with BCH-8 and BCH-16 are $10^5 \sim 10^8$ times as high as the error rate of raw DRAM even without ECC support.

For those reasons, 4LC-PCM needs an ECC scheme more effective than BCH-16, for

example, BCH-24 or BCH-32. However, the use of BCH-24 and BCH-32 is limited to devices that are lenient to timing delay and designed to operate at a relatively low data rate. For example, since MLC-NAND based devices delivers only a few tens of megabytes per second, and they are not sensitive to latency, BCH-24 or BCH-32 can be effectively implemented into them. However, 4LC-PCM as main memory of a system is sensitive to latency and delivers more than a few gigabytes per second. Thus, a complex ECC mechanism, like the BCH-24 and BCH-32, is not a suitable solution for 4LC-PCM considering the cost and performance problems. In light of the cost problem, applying complex ECC to a memory controller is not desirable because the current industry trend fabricates a memory controller and a processor core on the same die, which requires a separate CPU that supports 4LC-PCM. In light of the performance problem, the large computational overhead stemming from complex ECC compromises the performance in exchange for the reduced error rate and deteriorates the memory latency. In the sense, a typical DRAM system only implements simple ECC mechanisms, such as (72,64) Hamming code. We argue that using a complex and strong ECC mechanism does nothing but limiting the application of PCM and cannot render 4LC-PCM practically feasible for main memory.

5.3 Half-and-Half Storage: Improving Error Resiliency of Approximate Solid-State Memory by Co-Locating Precise and Approximate Information

5.3.1 Background

With the increasing concerns of power and energy in today's computing systems, *approximate computing* draws significant attention as one of the promising ways for energy-efficient computing [104, 105, 106, 107, 108, 109, 110]. Soft errors are unbearable in general, but certain categories of applications, such as multi-media processing and computer vision, can tolerate some amount of soft errors while minimizing output quality loss. As such, approximate computing trades off accuracy for energy and performance using software and hardware techniques.

With the same objective of energy efficiency, non-volatile memory such as phase change memory (PCM), spin-transfer torque RAM (STT-RAM), and memristors has also recently received significant attention as a replacement for DRAM. The domain of approximate computing can be extended to such non-volatile memory to provide more energy-efficient memory systems. For example, Sampson et al. [106] recently proposed relaxing the repeated write-and-verify sequences of a multi-level-cell (MLC) PCM write when storing approximate data.

Although approximate computing embraces imprecision, however, it is crucial to streamlining error resilience for the best trade-off between accuracy, performance, and energy. The same holds true for approximate storage as well. This chapter provides a comprehensive study to efficiently enable MLC PCM as approximate storage. We show that simply reducing the number of write iterations for approximate MLC PCM does not provide good error-resilient approximate storage.

We then propose a new type of multi-level PCM cells for approximate storage, which we refer to as a "half-precise and half-approximate" cell. To do so, we shift the resistance range of the second storage level (L2) in 4LC PCM to the lower resistance level (L1) and thus create *non-equispaced* storage levels. The proposed writing strategy, combined with Gray coding, makes the most significant bit in a four-level-cell (4LC) PCM precise without compromising write latency and energy, thereby having the great potential to improve computational resilience to errors in the context of approximate storage.

5.3.2 Multi-Level-Cell Phase Change Memory as Approximate Storage

5.3.2.1 Phase Change Memory (PCM)

Phase change memory (PCM) is a type of non-volatile memory that stores information as a resistance value. For example, a single-level PCM cell stores one bit of information (*i.e.,* zero or one) in two different resistance states: an amorphous state (high resistivity; reset) and a crystalline state (low resistivity; set). When a PCM cell is in a set state, its resistance range is around a few kilo-ohms, while the resistance range of the reset state

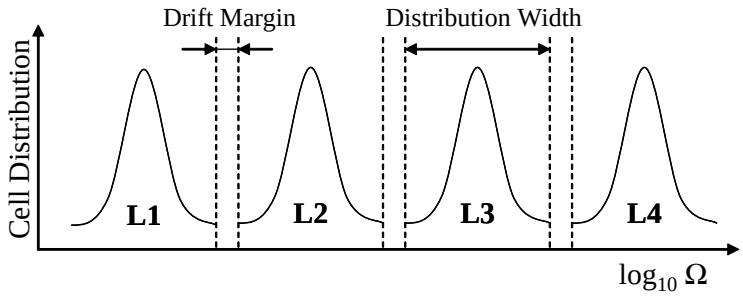


Figure 38: Write probability of a multi-level PCM cell. MLC PCM can either be precise or approximate depending on the distribution width of each storage level.

is around a few mega-ohms. Because of the large difference in resistance between the two states (three orders of magnitude), researchers have proposed multi-level-cell (MLC) PCM that defines intermediate storage levels between the set and reset states to increase information density in a PCM cell [91, 111]. For example, Figure 38 shows four-level-cell (4LC) PCM in which the resistance ranges of four storage levels are evenly distributed in a log-scale manner; e.g., each storage level targets the resistance range of 1kΩ, 10kΩ, 100kΩ, and 1MΩ. Unfortunately, PCM writes are non-deterministic; thus, a PCM write targeting 10KΩ may end up making PCM to have a resistance of only 5KΩ for instance. Therefore, MLC PCM needs to repeatedly perform a write-and-verify sequence until the write has been performed within a pre-defined resistance range (distribution width in Figure 38) of a storage level.

5.3.2.2 Precise and Approximate MLC PCM

Due to the nature of PCM materials, the resistance programmed in a PCM cell increases over time. This phenomenon, referred to as *resistance drift*, does not cause soft errors in single-level-cell (SLC) PCM; SLC PCM always returns the value initially written to. In contrast, MLC PCM is inherently *approximate storage* as the resistance drift can cross the decision boundary between code words (e.g., 00, 01, 11, 10 in 4LC PCM); thus, it may return a value different from the one initially stored in a cell after a few minutes since writing. To alleviate the drift-induced soft errors, there must be a large drift margin (guard

band) between the storage levels; that is, a multi-level cell can be precise or approximate by controlling the drift margin/distribution width of storage levels.

When PCM is used as main memory as a replacement for DRAM, it is expected to be as reliable as DRAM. Thus, we define precise MLCs as multi-level cells whose bit-level error rates are comparable to DRAM. Most of previous studies on 4LC PCM use the distribution width of $log_{10}R = 0.91\dot{6}$ that leads to 1000ns of PCM write latency. These 4LC PCMs are in fact already approximate storage by the standard; *i.e.,* for the distribution width, both MSB and LSB have non-negligible error rates as shown in Figure 39a. We use this error-prone 4LC PCM as baseline approximate 4LC PCM in this chapter.

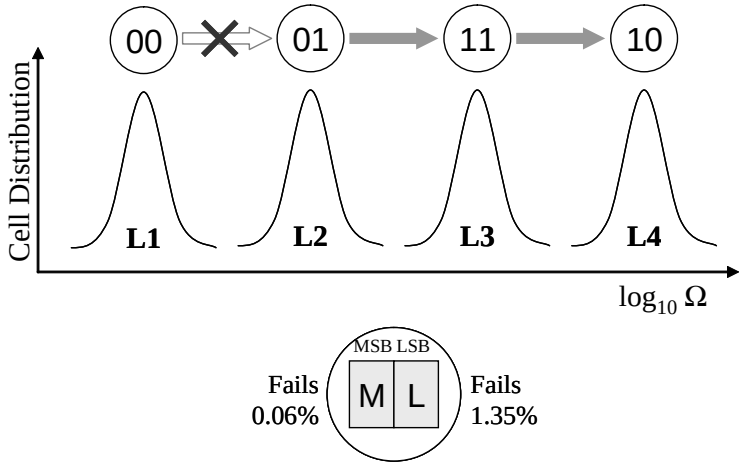
5.3.2.3 The Need for Reliable Approximate Storage

Prior work discussing approximate MLC PCM [106] exploits the relationship between the distribution width and the number of write iterations; *i.e.,* approximate data is written to the PCM cells with reduced drift margins to improve the write latency and energy. However, simply relaxing a write-and-verify sequence in cell programming does not enable efficient and reliable approximate MLC PCM. Unfortunately, such an approximate PCM cell would have non-negligible errors in bits of a PCM cell due to resistance drift. As we will discuss more in detail in Section 5.3.4, the key to enabling effective approximate MLC is to provide reliable high-order bits. In the next section, we discuss the writing strategy to provide more error resilient approximate PCM.

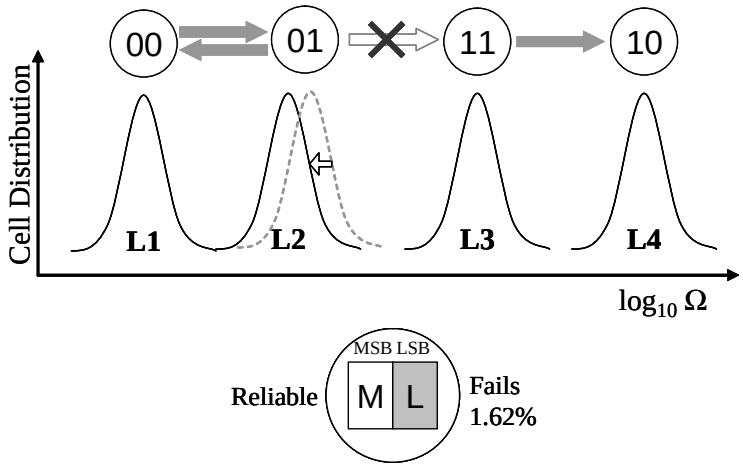
5.3.3 Half-and-Half PCM

5.3.3.1 Overview

Each storage level in approximate 4LC PCM has a unique error rate. For example, the first (L1) and the last (L4) storage levels do not generate errors, whereas the second (L2) and the third (L3) storage levels have 0.25% and 5.39% error rates after 45 minutes of initial writes due to resistance drift. For the 4LC PCM, if we convert the *storage-level* error rates into *bit-level* error rates, the first bit (MSB) and the second bit (MSB) have error rates of



(a) Approximate 4LC PCM (conventional)



(b) Proposed Half-and-half PCM

Figure 39: Half-and-half storage PCM secures reliability of the MSB by compromising error rates for LSB

0.06% and 1.35%, as shown in Figure 39a.[5] While mapping the highest-order bits of a value to the MSB of PCM cells [106] may improve error resiliency of approximate MLC PCM compared to a conventional PCM bit mapping, it can still lead to huge errors due to the non-negligible error rates of the MSB (see Section 5.3.4).

To provide an approximate multi-level cell that is more resilient to soft errors than the baseline approximate cell, we leverage the fact that one can write at any arbitrary resistance level on a PCM cell without compromising write latencies [112]. In fact, the equispaced

[5]We assume that the chances of appearance of all code words (00, 01, 11, 10) are the same.

resistance ranges of L1~L4, as illustrated in Figure 39a, are simply used because the configuration yields the lowest average bit-level error rates. However, as previously discussed, approximate storage that provides a number of precise bits (even though the rest of the bits are more compromised) is more beneficial in many cases than the one with lower average error rates (but no precise bits provided). As such, we propose to shift the second storage level (L2) to a lower resistance level, as illustrated in Figure 39b, thereby increasing the gap between L2 and L3. When such a simple change is combined with Gray code (00, 01, 11, and 10 for L1, L2, L3, and L4, respectively), commonly used for PCM cell encoding, the most significant bit can become error-free since we can eliminate the error sequence from 01 to 11. This way, we can have much reliable approximate cells for approximate data. Note that although this configuration may encounter errors between L1 and L2, which are not generated in the conventional 4LC PCM configurations, these errors do not affect the information stored in the MSB; Only the data stored in the LSB may be compromised. Also, the proposed half-and-half PCM has the same writing latency/power as conventional approximate MLC PCM.

5.3.3.2 Error Rates of Half-and-Half PCM

In this section, we compute error rates of the proposed half-and-half storage. We first determine the resistance range of the second storage level (L2) that does not generate errors between L2 and L3. For the discussion, we use the same analytical models and physical parameters as used in prior work [92, 93, 113]. We also conservatively assume that shifting the second storage level to the lower level does not improve the resistance drift rate.[6]

MSB Error Rates: Table 8 shows the error rates of the second storage level of 4LC PCM. The first column represents the elapsed time since initial writing, and the second column shows the error rates of the baseline resistance level, which is $\log_{10} R = 4.0$. The last two columns show the error rates when we slightly move L2 to the lower resistance

[6]Our modification moves L2 to the lower resistance level, which will decrease (or improve) the drift rate. This will only improve the LSB error rate of half-half PCM.

Table 8: Error rates for the second storage level (L2) of 4LC PCM

Elapsed Time	Original	$\log_{10} R = 3.9$	$\log_{10} R = 3.8$
5 minutes	0.09%	3.82×10^{-8}%	(too small)
15 minutes	0.15%	8.50×10^{-6}%	(too small)
25 minutes	0.19%	4.53×10^{-5}%	(too small)
35 minutes	0.22%	1.13×10^{-4}%	(too small)
45 minutes	0.24%	2.07×10^{-4}%	3.53×10^{-12}%

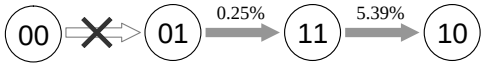
levels of $\log_{10} R = 3.9$ and $\log_{10} R = 3.8$. We mark "(too small)" when Mathematica 8.0 cannot compute the value because of lack of precision. In addition, a darker background cell indicates that the bit-level error rate is lower than that of DRAM. As shown in the table, when the resistance level of L2 is moved from $\log_{10} R = 4.0$ to $\log_{10} R = 3.8$, the error sequence of 01→11 is negligible; *i.e.,* the most significant bit of a MLC PCM cell becomes as reliable as a DRAM cell.

LSB Error Rates: We now discuss the impact of the half-and-half configuration on the LSB error rate. At a high level, the LSB of half-and-half PCM would intuitively have a higher error rate than conventional 4LC PCM because the proposed configuration causes soft errors between L1 and L2 in addition to the existing errors between L3 and L4. The LSB errors by L1 and L2 are in fact broken into the two different types of errors. First, the first storage level (L1) now causes drift-induced errors since the decision boundary between L1 and L2 would also be shifted to the lower resistance level when we use the resistance level of $\log_{10} R = 3.8$. Second, since we simply shift L2's distribution function while using the same writing methodology/precision as in conventional 4LC PCM, the new decision boundary now may generate initial writing errors; *i.e.,* the attempts to writing to L2 may accidentally end up writing to L1. As such, to compute the overall error rates of the LSB, we evaluate these two types of errors and add them together.

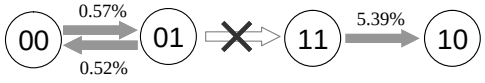
Table 9 shows the error rates of the first level (L1) for a half-and-half PCM cell. The second column shows the initial writing error rate, and the third column shows the drift-induced error rate that is a function of elapsed time. All in all, after 45 minutes of initial

Table 9: Error rates of the first storage level (L1) for half-and-half 4LC PCM

Elapsed Time	Initial errors (=A)	Drift-induced errors (=B)	Combined (=A+B)
5 minutes	0.52%	0.03%	0.56%
15 minutes		0.04%	0.56%
25 minutes		0.04%	0.57%
35 minutes		0.05%	0.57%
45 minutes		0.05%	0.57%



(a) Approximate 4LC PCM (conventional)



(b) Proposed Half-and-half PCM

Figure 40: Error diagram for half-and-half storage.

writing, about 0.57% of the L1 cells are falsely read out as L2. The L2 error rate (01→00) can be simply calculated because the L2 error rate is the same as its initial writing error rate (L2 does not cause drift-induced errors). Because all the storage levels have the same distribution function, the initial writing error rate of L2 is the same as that of L1; *i.e.*, 0.52% of L2 will be falsely read out as L1.

Comparison to Conventional 4LC PCM: Figure 40 shows the summary of the storage-level error rates of the proposed half-and-half storage; after 45 minutes since writing, 0.57% of L1 moves to L2, 0.52% of L2 moves to L1, and 5.4% of L3 moves to L4. Table 10 shows the bit-level error rates of both conventional approximate 4LC PCM and half-and-half PCM, which are converted from the storage-level error rates in the same manner as previously discussed. Again, a dark background cell indicates that the error rate is lower than that of DRAM.

As shown in Table 10, the key difference between these two technologies is that the proposed technique guarantees the reliability of MSB while the other does not. In exchange

Table 10: Bit-level error rates of two approximate PCM cells: 4LC PCM and half-and-half PCM

	Four-Level Cell PCM		Half-and-Half PCM	
Time (min)	MSB	LSB	MSB	LSB
5	0.02%	0.51%	(too small)	0.78%
10	0.03%	0.72%	(too small)	0.99%
15	0.04%	0.86%	(too small)	1.13%
20	0.04%	0.97%	(too small)	1.25%
25	0.05%	1.07%	(too small)	1.34%
30	0.05%	1.15%	(too small)	1.42%
35	0.06%	1.22%	(too small)	1.49%
40	0.06%	1.29%	(too small)	1.56%
45	0.06%	1.35%	8.83×10^{-14}%	1.62%

for such a benefit, half-and-half PCM compromises (1) LSB error rates and (2) average bit-level error rates of both MSB and LSB. However, we will show in subsequent sections that even though half-and-half PCM exacerbates errors on LSB and average bit-level error rates, it significantly improves robustness of stored values than the traditional PCM.

5.3.4 Bit-Level Errors to Value Errors

Bit-level errors in storage systems lead to value errors; however, each bit error has different impact on the value of the stored data. In some extreme cases, a single-bit error in a double-precision variable can change the stored value up to 3.5×10^{618}, or in the other extreme cases, the error may only change the value as little as 1.0×10^{-300}. Therefore, storing the most important piece of information in a place with the least error is important to minimize errors of stored values. Sampson *et al.* [106] recently proposed a simple coding scheme for approximate MLC PCM that minimizes value errors. This section first discusses the coding scheme and shows that only a single-bit error of conventional MLC PCM can largely compromise the robustness of the storage system. We then show that the proposed half-and-half PCM can significantly improve computational resilience to errors.

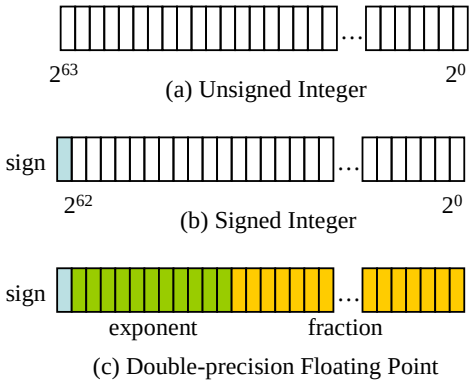


Figure 41: Bit mapping for (a) unsigned integer, (b) signed integer, (c) double-precision floating-point (IEEE 754)

5.3.4.1 Assigning Binary Values to Multi-Level Cells

Sampson *et al.* [106] examined two different codes for assigning binary values to MLC PCM; concatenation and striping code. Concatenation code assigns n consecutive binary bits to an n-bit cell, whereas striping code assigns first n bits to n different cells. The striping code basically exploits lower error rates of MSB in MLC PCM and stores important information in the MSB and shows a better error tolerance. Therefore, we assume that the baseline coding scheme is striping code where the first $n/2$ bits are stored on MSB while the lower bits are stored on LSB of n PCM cells. This coding applies for both the traditional 4LC PCM and half-and-half PCM.

5.3.4.2 Impact of Single-Bit Errors

Bit flipping in storage value errors for virtually any data type including (1) integers and (2) floating-point types.

Unsigned Integer: Due to its simplicity, we first discuss impact of a single-bit error on an integer type of data. Figure 41a shows a typical bit-mapping for an unsigned integer where the nth bit from the LSB represents 2^{n-1}; thus, a bit flip on the nth bit leads to a value error of 2^{n-1} in this case. If we define $E(n)$ as the expected error rate of the nth bit, then the expected value error for an m-bit unsigned integer becomes $\sum_{n=1}^{m} E(n) \times 2^{n-1}$. Thus, the best mapping strategy is clearly to assign the least failing bit from the most significant

bit. Surprisingly, with this simple but optimal mapping strategy, in only five minutes after writing, a 64-bit unsigned integer in conventional approximate 4LC PCM is expected to have a value error of 4.01×10^{15}. Here we use 32 4LC PCM cells for mapping, and the MSBs of the 32 cells store the 32 high order bits of a 64-bit integer.

On the other hand, the same integer data type using half-and-half PCM is expected to have a value error of 3.35×10^{7}, which is about 10^{8} smaller compared to conventional 4LC PCM. For a 32-bit unsigned integer, conventional approximate 4LC PCM shows the expected value error of about 934,316.9, whereas half-and-half PCM only shows 255.6. Thus, for both 64-bit and 32-bit cases, half-and-half PCM shows several orders of magnitude less value errors compared to conventional 4LC PCM.

Signed Integer: Signed integer types also show the same amount of expected value errors as the unsigned integer types when two's complement representations are used. Signed integer types (Figure 41b) use the first bit to indicate whether the value is positive or negative; therefore, the value error depends on the rest of the bits. However, when signed integer types employ two's complement representations, it is easy to analyze the impact of an error on the sign bit. In two's complement system, when sign bit becomes zero (positive) from one (negative), the value of such integer is subtracted by 2^{m-1} where m is the number of bits in the integer. For example, we consider an eight-bit signed integer variable with the stored value of three, then its binary representation is $0000\ 0011_b$. In the case of sign bit error, it becomes $1000\ 0011_b$, or -125 in two's complement representation. The amount of value error in this case is 128 or $2^7 = 2^{m-1}$. This amount of error is exactly the same as we found from the unsigned integer types; therefore, we argue that the same analysis still holds for signed integers. In summary, we find that for the two's complement representation, the accuracy benefit of the proposed half-and-half PCM also holds for signed integer types.

Table 11: Bit flipping happens on π stored in double-precision floating point

Error Position	Bit Layout	Value
(no error)	0x4009 21FB 5444 2D18	3.1416 (=π)
48th bit	0x4009 A1FB 5444 2D18	3.2041
49th bit	0x4008 21FB 5444 2D18	3.0166
50th bit	0x400B 21FB 5444 2D18	3.3916
51st bit	0x400D 21FB 5444 2D18	3.6416
52nd bit	0x4001 21FB 5444 2D18	2.1416
53rd bit	0x4019 21FB 5444 2D18	6.2832
54th bit	0x4029 21FB 5444 2D18	12.566
55th bit	0x4049 21FB 5444 2D18	50.265
56th bit	0x4089 21FB 5444 2D18	804.25
57th bit	0x4109 21FB 5444 2D18	2.06×10^{5}
58th bit	0x4209 21FB 5444 2D18	1.35×10^{10}
59th bit	0x4409 21FB 5444 2D18	5.80×10^{19}
60th bit	0x4809 21FB 5444 2D18	1.07×10^{39}
61st bit	0x5009 21FB 5444 2D18	3.64×10^{77}
62nd bit	0x6009 21FB 5444 2D18	4.21×10^{154}
63rd bit	0x0009 21FB 5444 2D18	1.27×10^{-308}
64th bit	0xC009 21FB 5444 2D18	-3.1416

Floating Point: In general, floating point data types are more common and important than integers in approximate computing domains. The expected value error of a floating-point variable depends on the value initially stored in approximate storage. For example, assume that $\pi(= 3.141592...)$ is stored in a 64-bit double-precision data type and that a bit-flipping error happens on the 51st bit. In this case, the absolute error (|initial value − altered value|) becomes 0.5. However, if the initial value is 2π, the absolute error for the same bit flip becomes 1.0; thus, it is not trivial to define and quantitatively compare the expected value errors across different approximate storage. However, we can still compare the expected value error when we fix the initial value with one of the widely used constants and show that the proposed half-and-half PCM provides many orders of magnitude less value errors than the traditional PCM.

Table 11 shows the changes in values by the location of a single-bit flipping error when *pi* is stored in a 64-bit double-precision variable. The errors on 53rd through 64th bits result

in more than 100% of absolute errors, and the most significant error shows the absolute error of more than 10^{150} when the 62nd bit is flipped. In contrast, the maximum value error of half-and-half PCM is 9.54×10^{-7}. For other constants, we have observed that half-and-half PCM are similarly better than conventional 4LC PCM.

5.3.4.3 Approximate 4LC PCM with Error Correcting Codes

Bit-level errors can be detected and corrected using error correcting codes (ECC), so one interesting question might be the possibility of using approximate 4LC PCM with ECC to improve the error resiliency of approximate storage. However, using ECC is a less appealing solution in approximate computing than in conventional computing. One main reason is the overhead of ECC. The main purpose of using approximate storage is to improve performance/energy. However, ECC will introduce extra storage overhead or another dedicated chip that drives signals for increased numbers of data lines. Memory controller must also occupy extra space, consume latency, and burn extra power for encoding, decoding, and correcting errors for all the transferred data. In contrast, the proposed half-and-half PCM does not incur extra area, latency nor power overhead compared to approximate 4LC PCM with ECC.

5.3.5 Costs of Writing Precise Bits in 4LC PCM

4LC PCM can be as reliable as DRAM if we reduce the distribution width and increase the guard band. Here, we discuss how narrow the distribution width needs to be to make the 4LC PCM precise. For the discussion, we use the equations from other study (Equations (5) and (6) in [113]) and use the distribution width of $log_{10}R = 0.91\dot{6}$ as baseline (100%).

Table 12 shows the error rates of MSB and LSB when we reduce the distribution width from 100% to 40% as illustrated in Figure 42. Cells with darker background indicates that the error rates are comparable to or lower than those of DRAM. As shown in the table, the MSB starts to be as reliable as DRAM from 60% of the baseline distribution width, whereas the LSB begins reliable around the half of the original width. Then, the next question is

Table 12: Bit-level Error Rates of MSB and LSB by the width of the resistance range

Distribution width	MSB	LSB
100%	0.06%	1.35%
90%	0.01%	0.61%
80%	5.07×10^{-04}%	0.17%
70%	1.75×10^{-06}%	0.02%
60%	4.47×10^{-10}%	3.05×10^{-4}%
50%	7.57×10^{-15}%	5.59×10^{-7}%
40%	(too small)	5.38×10^{-11}%

how many write iterations we need to halve the distribution width.

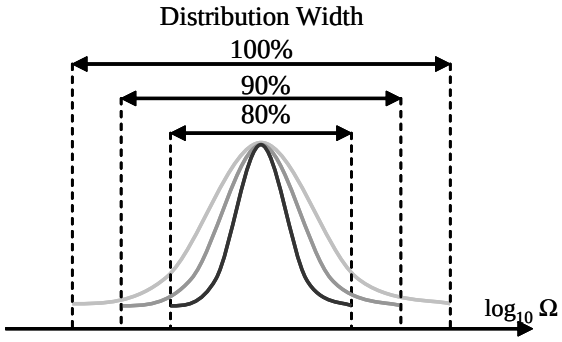


Figure 42: Shrinking distribution width of MLC PCM

Roughly speaking, halving the distribution width would increase the number of write iterations as similar as that is required for doubling the numbers of storage levels in MLC PCM. Assume that one decides to write a 4LC PCM cell with half of the distribution width. In this case, one can either (1) define extra four storage levels between existing four to create a 8LC PCM cell or (2) leave extra storage levels empty as drift margins. Because the writing precision remains the same for both cases, one should expect the same numbers of write iterations as well. Thus, we can compute the number of required write iterations to halve the distribution width for 4LC PCM by calculating the average write iterations that 4LC and 8LC PCM (distribution width of $log_{10}R = 0.91\dot{6}$) takes.

Figure 43 shows the number of write iterations required for 4LC and 8LC PCM. On average, writing on 4LC PCM takes 8.7 iterations, whereas writing on 8LC PCM takes 19.3

Table 13: Bit-level error rates and write latencies

Technology	Error Rates			Write Latency
	MSB	LSB	(Avg)	
Baseline	0.06%	1.35%	0.71%	1000ns
60%	-	$3.1E^{-4}$%	$1.5E^{-4}$%	1667ns
50%	-	-	-	2000ns
Half-and-half PCM	-	1.62%	0.81%	1000ns

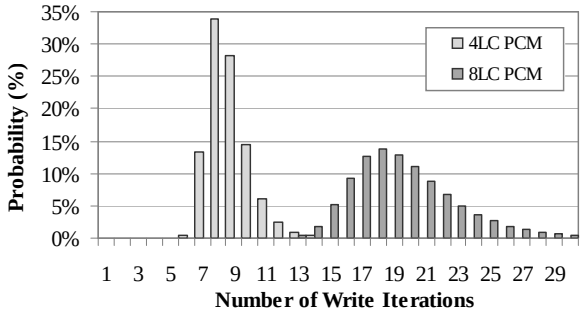


Figure 43: Distribution of the number of write iterations for 4LC and 8LC PCM

iterations (about 2.2x). Another interesting change for 8LC PCM is that it has a longer tail than 4LC, which can degrade the worst case performance of writing PCM. We assume that other techniques [114] can mitigate such side effects and simply use the average number of write iterations. Then, we can estimate the cost of writing two precise bits on 4LC PCM as 2000ns, one precise MSB and one approximate LSB as 1667ns (= $1000ns/60\%$), and two approximate bits as 1000ns (= $1000ns/100\%$). Table 13 summarizes the write latencies compared to half-and-half PCM.

5.3.6 Evaluation

5.3.6.1 Sensitivity study for half-and-half PCM

The proposed half-and-half PCM in Section 5.3.3 relocated the center of the resistance distribution (=μ_R) of the second storage level from $\log_{10}\mu_R = 4.0$ to $\log_{10}\mu_R = 3.8$, which made the MSB of it reliable. $\log_{10}\mu_R = 3.8$ is an optimal point for the given number of write iterations or write latency of $1000ns$. However, we have shown that the error rate of

Table 14: MSB Error Rates for Half-and-half PCM with Relaxed Write Iterations by μ_R of L2

Dist. Width	$\log \mu_R = 3.78$	$\log \mu_R = 3.74$	$\log \mu_R = 3.70$	$\log \mu_R = 3.66$	$\log \mu_R = 3.62$
110%	$2.66E^{-10}$%	$2.34E^{-14}$ %	(too small)	(too small)	(too small)
120%	$4.32E^{-9}$%	$3.72E^{-10}$ %	$4.37E^{-14}$ %	$6.20E^{-19}$ %	(too small)
130%	$5.45E^{-7}$%	$4.05E^{-7}$ %	$4.60E^{-10}$ %	$7.25E^{-14}$ %	$1.48E^{-18}$ %
140%	$7.25E^{-4}$%	$3.53E^{-5}$ %	$3.38E^{-7}$ %	$5.10E^{-10}$ %	$1.07E^{-13}$ %
150%	$1.98E^{-3}$%	$3.49E^{-4}$ %	$2.06E^{-5}$ %	$2.52E^{-7}$ %	$4.99E^{-10}$ %

LSB is less sensitive to value errors as long as MSB is reliable, and there are cases where write latency is more important than the error rate of LSB. In other words, half-and-half PCM can relax on write iterations or reduce write latency by further sacrificing the error rate of LSB while still maintaining the most important property of it; reliable MSB.

To examine the relationship between write latency and the error rate of LSB, we first evaluate the impact of stretching the distribution width of the second storage level. Starting from the original half-and-half configuration, we stretch the distribution width from 100% (=$log_{10}R = 0.91\dot{6}$) to 150% (=$log_{10}R = 1.375$) in the step of 10%. For all cases, μ_R of L2 is moved toward L1, and μ_R of L3 is moved toward L4 for the same amount so that MSB is still reliable. Note that as we have wider distribution width of storage levels, we must further move L2 and L3 toward L1 and L4 respectively, and this relocation will compromise error rates of LSB.

We first examine how much μ_R of L2 must be relocated toward L1 to have no MSB errors for first 45 minutes. For each distribution width from 100% to 150%, we start from the original configuration, $\log_{10} \mu_R = 3.80$, and move μ_R toward L1 until it shows no errors between L2 and L3. When the distribution width is 110% and 150%, we had to move μ_R to $\log_{10} \mu_R = 3.78$ and $\log_{10} \mu_R = 3.62$ respectively. This relation is summarized in Table 14 where darker backgrounds indicate the error rates less than that of DRAM.

Table 15: Error Rates of Half-and-half PCM with Relaxed Write Iterations

Dist. Width	L1→L2 Error	L2 Initial Error	L3→L4 Error	Combined LSB Error	Write Latency
110%	0.95%	0.84%	6.89%	2.17%	909 *ns*
120%	1.42%	1.27%	10.35%	3.26%	833 *ns*
130%	1.96%	1.77%	14.80%	4.63%	769 *ns*
140%	2.62%	2.38%	20.35%	6.34%	714 *ns*
150%	3.45%	3.14%	26.96%	8.39%	667 *ns*

Now for the given distribution width and μ_R, we calculate error rates for LSB. As discusses earlier, LSB error is a function of (1) errors from L1, which is the sum of the resistance drift error and initial writing error, (2) initial writing errors from L2 where the write attempt to L2 can write to L1, and (3) errors from L3 due to the resistance drift. Each type of errors are evaluated and presented in the second through fourth column of Table 15. We then show combined error rate of LSB and the expected write latency of each configuration. LSB experiences about 5.3 times more errors than the original half-and-half PCM as we stretch the distribution width from 100% to 150%. The remainder of this section examines the impact of increased error rates for LSB to the output quality of applications.

5.3.6.2 Benchmarks and Definition of Output Quality Loss

We evaluate all SciMark2 benchmarks, Fast Fourier Transform (FFT), Jacobi Successive Over-relaxation (SOR), Monte Carlo π calculation (MCπ), sparse matrix multiply (SMM), dense LU matrix factorization (LU) from EnerJ [115]. For each benchmark, we define the output quality loss as follows.

- **FFT:** FFT takes a linear array size of *n* and Fourier transform the array. We first perform Fourier transform to the input array and also apply inverse Fourier transform to the output and compare it against the original array. Error scale is the same as LU.
- **SOR:** SOR takes a 2D matrix of *n* by *m* and write its computational output to the matrix itself. We copy the input matrix and inject errors to the original one. In addition, both matrices are processed by SOR and the results are compared. Error scale is the same as

LU.

- **MCπ:** MCπ generates two random doubles and calculate sum of square of each double. By repeatedly doing so, MCπ calculates π. This experiment assumes that reading the calculated sum of two doubles generate reading errors. The output quality is defined as difference between calculated π from perfect reading versus calculated π from erroneous reading.
- **SMM:** SMM from SciMark2 employs compressed-row format and a prescribed sparsity structure. This experiment assumes that reading the compressed structure generates errors. Output quality metric compares multiplied matrices element by element in the scale of 0 to 1. Overall quality of output is average of scale of all elements.
- **LU:** LU takes *n* by *n* matrix and output another *n* by *n* matrix. We compare the output matrices element by element and scale the difference from 0, no quality loss or identical, to 1, totally different. This scale is an absolute value of difference divided by the results from the precise run. If it is zero, then the scale becomes the difference. The scale cannot exceed 1. The output quality loss for LU is an average of scales of all elements.

5.3.6.3 Evaluation Methodology

Our usage scenario assumes reading PCM cells 5 ~ 45 minutes after the initial writing. Because simulating computer systems for tens of real time minutes requires prohibitive computing power or time, we present the following methodology. We first divide the entire memory footprint of a benchmark into two categories; (1) the storage for input data and (2) the storage for by-products or output data. In addition, we inject MSB and LSB errors for the read accesses to the category (1) while guarantee the perfect read / write accesses for (2). For example, LU, one of benchmarks from EnerJ [115], takes *n* by *n* matrix as an input and calculates another *n* by *n* matrix after decomposing the matrix into lower and upper parts. In such a case, the input matrix becomes the category (1) in our case while the rest

of the memory footprint becomes (2). The rationale behind this setup is that because we only consider read errors for long-term writes, the input data or category (1) is the only part that falls into this criteria. All other memory footprint including intermediate, temporary variables, and output matrix is being written and reused almost immediately.

We evaluate impact of MSB and LSB error rates to the quality of output by natively running benchmarks. The quality of output is a metric of how similar the approximate and precise results are, but not about the performance. Therefore, we can safely skip micro-architectural simulations and run the benchmarks and error injectors on a native machine without compromising the correctness of the experiment. Error injectors consume CPU time and memory footprint; however, they do not change the outcome of the benchmarks. Moreover, simulating bit-level errors using micro-architectural simulators is not practical for the following reason. Because error injectors roll a dice every time they need to generate errors, the outcome of the results of our experiments is naturally non-deterministic. Therefore, we have to repeat running the same benchmark over hundreds of times to reach a stable data point, which takes hours in some cases. Simulating hours of native run using simulators is impractical especially when we need the actual calculated results where we cannot sample, skip, and fast-forward the simulation.

5.3.6.4 Experimental Results

We evaluate the impact of error rates in Table 10, bit-level error rates of 4LC and half-and-half PCM, to the output quality. Figure 44 presents output quality loss of the baseline approximate 4LC PCM. In this experiment, we find the following observations. Firstly, output quality loss is a function of the size of the input matrix for all the benchmarks. We evaluated from a tiny 10 by 10 matrix to a large 200 by 200 for LU, from 256 to 2048 elements of an array for FFT, and from 20 by 20 to 80 by 80 matrix for SOR, to find out that the output quality loss increases with the size of the input. For example, right most markers from Figure 44e show the output quality loss after 45 minutes of initial writing. As we increase the input matrix size from 10 by 10 to 200 by 200, output quality loss

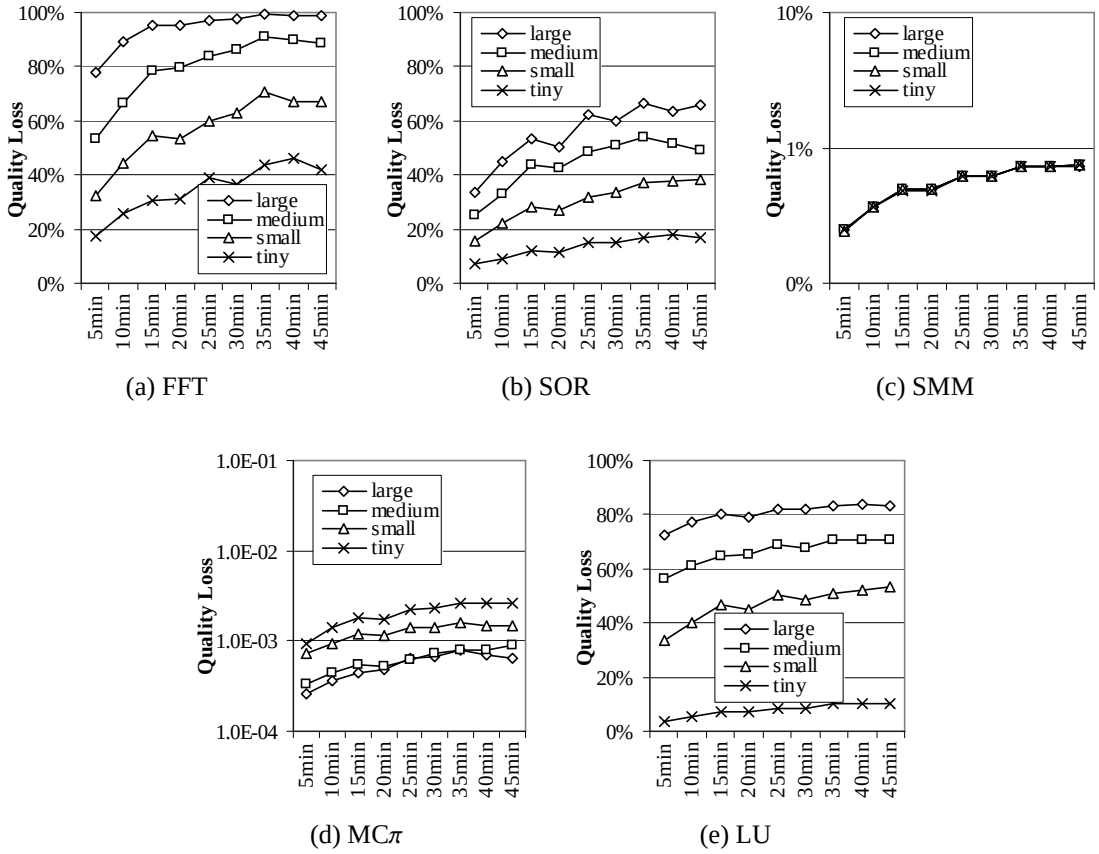


Figure 44: Output Quality Loss for Approximate 4LC PCM (conventional)

increased from about 10% to over 80%. This is because when the input matrix is big, errors easily propagate to the other cells of a matrix. When there are only ten elements in a row, an error on 9th element only propagate to the 10th cell; however, for a matrix of 200 elements, an error on 9th element propagates to the rest, 191 elements. Secondly, each benchmark shows different sensitivity on the output quality loss by the bit-level error rates. For example, for large input cases, quality loss in LU increased from 70% to 84% while for the SOR case, the output quality loss almost doubled. However, we also find that size of the workload shows more significant impact to the output quality loss.

Now we compare output quality loss of half-and-half PCM against the baseline as shown in Figure 45. As expected, quality loss of the proposed PCM was orders of magnitude less than the baseline for all the benchmarks tested. More specifically, for a large

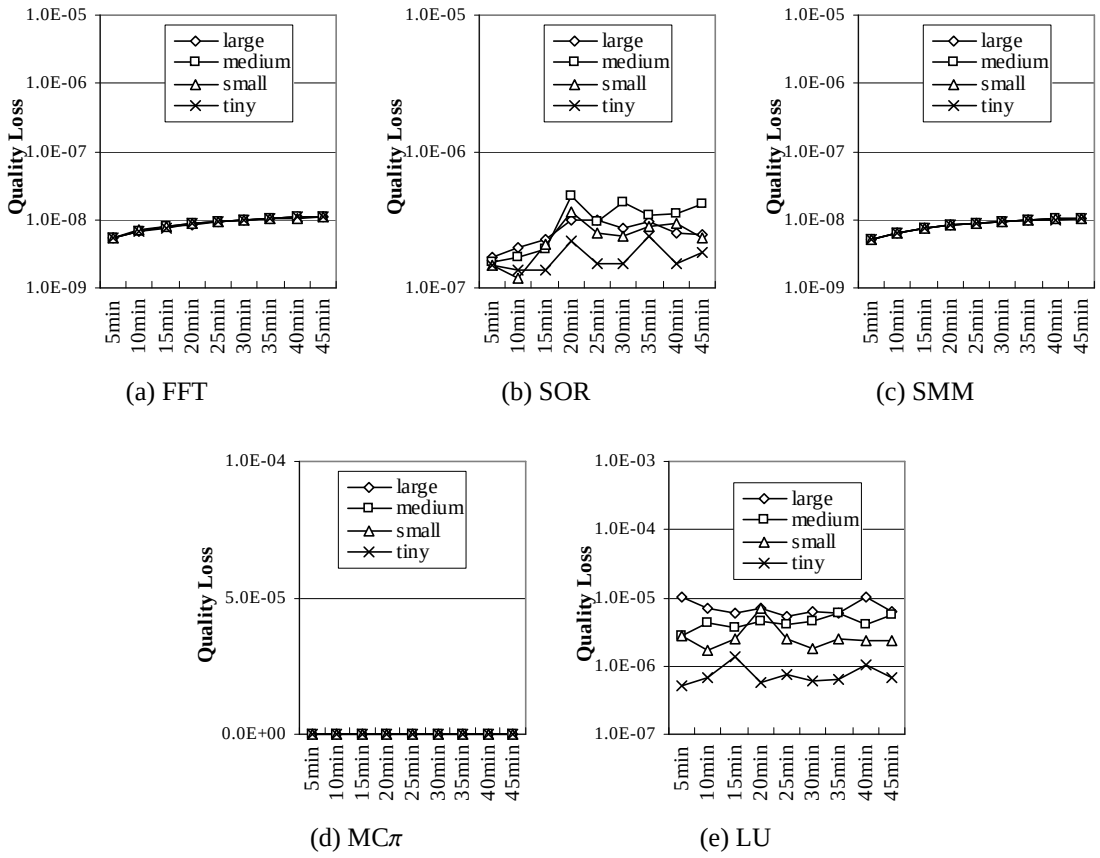


Figure 45: Output Quality Loss for Proposed Half-and-half PCM

matrix, output quality loss of half-and-half PCM for LU was constantly less than 10^{-5} while the baseline marked around 80%. For all other benchmark from EnerJ, we also find that the output quality loss of half-and-half PCM is orders of magnitude less than the conventional 4LC PCM.

5.3.7 Related work

Approximate computing basically trades accuracy for performance [107, 108]. Compromising barely noticeable accuracy in output may lead to orders of magnitude less power and energy consumption. Researchers proposed hardware techniques [116, 105, 117] including probabilistic CMOS (PCMOS) technology [109] while others proposed software techniques [118] or leveraged both the hardware and software techniques by exposing hardware control extensions to software [119].

While approximate computing mainly focuses on relaxing computational robustness, others examined approximation concept for storage systems. Error-tolerant part of memory footprint could be saved in less-frequently refreshed region of DRAM [110] or stored in non-volatile memory (NVM) with less power with improved latency [106]. Different from prior studies, we exploit a unique characteristic of MLC PCM, which could secure reliability of half of the information stored in a memory cell, to significantly improve resilience of the approximate storage.

Research community proposed several NVM technologies to mitigate the physical scaling challenges that DRAM face today. Among all emerging technologies, PCM is one of the most mature and promising technology in replacing DRAM as main memory [120, 121, 122, 123, 124]. Because the resistance of a PCM cell can be set at any arbitrary point between set and reset states, researchers found that defining more storage levels between set and reset states will result in storing more bits per cell or increasing the information density [111, 91]. However, the resistance level of a PCM cell increases over time, and such a drift generates soft errors [89, 90, 112]. To compensate errors induced by resistance drift, researchers proposed many techniques by leveraging data encoding and error correcting schemes [92, 95, 93, 94]. Other studies also examined a write time-aware scheme [96] or a smart scrubbing based scheme [92]. However, a recent work argued that MLC PCM is still requires architectural support to be as reliable as DRAM [125]. We, however, show that by exploiting resistance drift nature of error-prone MLC PCM, it can be configured as resilient approximate storage.

CHAPTER 6

CONCLUSION

This dissertation proposed various power optimization techniques for three different levels of datacenters; infrastructure level, system level, and micro-architecture level. An infrastructure-level study in Section 3.2 investigated resource provisioning properties of a heterogeneous cloud computing environment. Using mathematical models, Section 3.2 analyzed a perfectly parallelized task running on a heterogeneous cloud with distinct power efficiencies. To quantify the trade-off of resource provisioning, Section 3.2 used the energy-delay product as an objective metric to consider both performance and the utility consumption. To achieve an optimal EDP value, the expectation-based analysis showed that the response time ratio of the slowest node ($= b$) versus the fastest node ($= a$) must be less than or equal to three ($b/a \leq 3$). Findings suggest that computing nodes that are 3x or slower than the fastest node should be discarded from the cloud for achieving an optimal EDP. These models and analysis can be used to guide future deployment, allocation and upgrades of cloud infrastructure to achieve optimal utility effectiveness.

Another infrastructure-level study, SimWare, was presented in Section 3.3. Over years, researchers proposed to operate cooling units at a high discharge temperature to reduce cooling power. However, high room temperature can inadvertently lead to high fan rotation speed and eventually overwhelm the savings from the cooling units. To study and understand these compound effect, Section 3.3 presented a holistic simulator, SimWare, which simulates the detailed behavior of an entire datacenter. SimWare reports power and energy breakdown of a given datacenter by analyzing several critical components including the power of the servers and cooling units, the power of fans, the effect of heat recirculation, and the air-travel time for providing shrewd, effective decision in optimizing power, temperature, performance, and the operational cost. Experimental analysis using SimWare showed that much of the cooling efficiency is lost due to inlet air temperature differences

across servers.

This dissertation continued to a system-level power optimization technique, ATAC, in Chapter 4, which was motivated from observations made by SimWare. Section 4.2 began by carefully reviewing the fundamentals of datacenter cooling and found that considerable cooling energy is wasted because of (1) the safety margin that cooling units must ensure and (2) the non-uniform inlet air temperatures across servers. These issues stem from the location of each server relative to the CRAC unit and their height from the floor. To address this drawback, Section 4.2 proposed a system-level approach that first aggressively reduces the cool air supply from the CRAC unit to save power and then uses a new system-level control called ATAC, which is applied to each server. By sensing the inlet temperature to reduce the core temperature, ATAC can dynamically cap the performance of the server using DVFS. Using a modified SimWare framework with the Google production trace, Section 4.2 evaluated ATAC and found that a datacenter can reduce the cool air supply with 38% savings of cooling power, or 7% savings of total power while degrading performance by a negligible sub-1%.

Chapter 5 discussed micro-architectural techniques for power efficient datacenters under the context of emerging memory technologies. Section 5.2 showed that the error rate of 4LC-PCM cannot be reduced as low as the error rate of DRAM practically. Firstly, Section 5.2 introduced the mathematical model that estimated SER of MLC PCM, considering the following factors: (1) effect of resistance drift, (2) distribution functions of the resistance at $t_0 = 1s$, (3) distribution functions at the rate of resistance drift, and (4) effects of iterative writing mechanism. Secondly, Section 5.2 compared the results from the mathematical model to the results from Monte Carlo simulator for the purpose of validating the mathematical model. In addition, Section 5.2 used mean and deviation of distribution functions from other studies to show the relationship among the SER, scrubbing periods, and scrubbing overheads for 4LC PCM. Further analysis showed that 4LC PCM cannot be used as main memory given its high error rates and scrubbing overheads. The most critical

problem of 4LC PCM is high SER of the third storage level, which is about $10^9 \sim 10^{11}$ times higher than that of DRAM. With all in-depth analysis, due to resistance drift, 4LC PCM is either unreliable for practical deployment.

Section 5.3 examined error-prone 4LC PCM as approximate storage systems. Error-tolerant applications can utilize power efficient and high performance but approximate storage systems. Furthermore, when the computational results are consumed by human beings, such as rendered 3D images for video game users, errors in results can easily be justified. However, Section 5.3 argued that storing important pieces of information in a more reliable place with less errors significantly improved resiliency of approximate storage systems. Section 5.3, therefore, proposed a new class of MLC PCM cells by exploiting skewed and unevenly distributed storage levels for MLC PCM. This class of MLC PCM cells secured reliability of MSB while sacrifices reliability of LSB, *i.e.,* these cells are half-precise and half-approximate. Even though the average error rate is compromised, Section 5.3 showed that the proposed scheme significantly improved the quality of output. The proposed writing strategy also reduced writing iterations, power, and latency of the underlying memory technology while still achieved orders of magnitude more accurate results.

REFERENCES

[1] J. G. Koomey, "Growth in data center electricity use 2005 to 2010." http://www.analyticspress.com/datacenters.html, 2011.

[2] J. Kyathsandra and C. Rego, "High Ambient Temperature Data Center Efficiency and Intel." http://www.intel.com/content/www/us/en/data-center-efficiency/efficient-datacenter-high-ambient-temperature-operation-brief.html.

[3] RUMSEY Engineers, Inc., "Data Center Energy Benchmarking Case Study — Facility 8." Lawrence Berkeley National Laboratory (http://hightech.lbl.gov/dctraining/reading-room.html), 2003.

[4] M. LaMonica, "Yahoo opens doors to self-cooled data center." http://news.cnet.com/8301-11128_3-20016849-54.html.

[5] A. Rawson, J. Pfleuger, and T. Cader, "The Green Grid Data Center Power Efficiency Metrics: Power Usage Effectiveness and DCiE." The Green Grid, 2007.

[6] R. Ayoub, R. Nath, and T. Rosing, "Jetc: Joint energy thermal and cooling management for memory and cpu subsystems in servers," in *Proceedings of the 18th Annual Symposium on High Performance Computer Architecture*, HPCA-18, pp. 299–310, 2012.

[7] R. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[8] M. Tighe, G. Keller, M. Bauer, and H. Lutfiyya, "DCSim: A Data Centre Simulation Tool for Evaluating Dynamic Virtualized Resource Management," in *Proceedings of the 6th International DMTF Academic Alliance Workshop on Systems and Virtualization Management*, SVM '12, 2012.

[9] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making scheduling cool: Temperature-aware resource assignment in data centers," in *Usenix Annual Technical Conference*, pp. 61–75, 2005.

[10] R. Sharma, C. Bash, C. Patel, R. Friedrich, and J. Chase, "Balance of power: Dynamic thermal management for internet data centers," *IEEE Internet Computing*, pp. 42–49, 2005.

[11] Q. Tang, S. K. S. Gupta, and G. Varsamopoulos, "Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A

cyber-physical approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, pp. 1458–1472, 2008.

[12] S. Pelley, D. Meisner, T. Wenisch, and J. VanGilder, "Understanding and abstracting total data center power," in *Proceedings of the Workshop on Energy Efficient Design*, WEED '09, 2009.

[13] P. Bohrer, E. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony, "The Case for Power Management in Web Servers," *Power Aware Computing*, vol. 62, 2002.

[14] L. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[15] N. Tolia, Z. Wang, M. Marwah, C. Bash, P. Ranganathan, and X. Zhu, "Delivering Energy Proportionality with Non Energy-proportional Systems: Optimizing the Ensemble," in *Workshop on Power Aware Computing and Systems*, HotPower '08, USENIX Association, 2008.

[16] S. Pelley, D. Meisner, P. Zandevakili, T. F. Wenisch, and J. Underwood, "Power Routing: Dynamic Power Provisioning in the Data Center," in *Proceeding of the 15th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS-15, pp. 231–242, 2010.

[17] Micron, "512Mb DDR SDRAM (x4, x8, x16) Component Data Sheet." http://download.micron.com/pdf/datasheets/dram/ddr/512MBDDRx4x8x16.pdf, 2000.

[18] Rambus Inc., "128/144-MBit Direct RDRAM Data Sheet," May 1999.

[19] I. Hur and C. Lin, "A Comprehensive Approach to DRAM Power Management," in *Proceedings of the 14th International Symposium on High Performance Computer Architecture*, pp. 305–316, IEEE, 2008.

[20] D. Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: Eliminating Server Idle Power," in *Proceeding of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS-14, pp. 205–216, 2009.

[21] C. Lefurgy, X. Wang, and M. Ware, "Power capping: a prelude to power shifting," *Cluster Computing*, vol. 11, no. 2, pp. 183–195, 2008.

[22] D. H. Albonesi, "Selective cache ways: On-demand cache resource allocation," in *International Symposium on Microarchitecture*, pp. 248–, 1999.

[23] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power," in *Proceedings of the 28 th annual International Symposium on Computer Architecture*, vol. 29, pp. 240–251, 2001.

[24] K. Flautner, N. Kim, S. Martin, D. Blaauw, and T. Mudge, "Drowsy caches: simple techniques for reducing leakage power," in *Proceedings of the 29th Annual International Symposium on Computer Architecture*, pp. 148–157, 2002.

[25] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: a Low-power Pipeline based on Circuit-level Timing Speculation," in *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-36, pp. 7–18, 2003.

[26] D. Bull, S. Das, K. Shivshankar, G. Dasika, K. Flautner, and D. Blaauw, "A Power-efficient 32b ARM ISA Processor Using Timing-error Detection and Correction for Transient-error Tolerance and Adaptation to PVT Variation," in *Solid-State Circuits Conference Digest of Technical Papers*, ISSCC '10, pp. 284–285, 2010.

[27] G. Shamshoian, M. Blazek, P. Naughton, R. S. Seese, E. Mills, and W. Tschudi, "High-Tech Means High-Efficiency: The Business Case for Energy Management in High-Tech Industries." Lawrence Berkeley National Laboratory (http://hightech.lbl.gov/dctraining/reading-room.html), 2005.

[28] M. Jang, K. Schwan, K. Bhardwaj, A. Gavrilovska, and A. Avasthi, "Personal clouds: Sharing and integrating networked resources to enhance end user experiences," in *33rd IEEE International Conference on Computer Communications*, INFOCOM-33, 2014.

[29] H. Yoon, A. Gavrilovska, K. Schwan, and J. Donahue, "Interactive use of cloud services: Amazon sqs and s3," in *12th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, CCGrid, pp. 523–530, 2012.

[30] S. Yeo and H.-H. S. Lee, "Using mathematical modeling in provisioning a heterogeneous cloud computing environment," *Computer*, vol. 44, no. 8, pp. 55–62, 2011.

[31] L. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A Break in the Clouds: Towards a Cloud Definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2008.

[32] K. Xiong and H. Perros, "Service Performance and Analysis in Cloud Computing," in *Proceedings of the 2009 Congress on Services-I-Volume 00*, pp. 693–700, IEEE Computer Society, 2009.

[33] M. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon S3 for Science Grids: A Viable Solution?," in *Proceedings of the 2008 International Workshop on Data-aware Distributed Computing*, pp. 55–64, 2008.

[34] L. A. Barroso, "The Price of Performance," *ACM Queue*, vol. 3, no. 7, pp. 48–53, 2005.

[35] S. Ghiasi, T. Keller, and F. Rawson, "Scheduling for Heterogeneous Processors in Server Systems," in *Proceedings of the 2nd Conference on Computing Frontiers*, pp. 199–210, 2005.

[36] R. Nathuji, C. Isci, and E. Gorbatov, "Exploiting Platform Heterogeneity for Power Efficient Data Centers," in *Proceedings of ICAC'07: Fourth International Conference on Autonomic Computing*, 2007.

[37] PassMark, "CPU Benchmarks." http://www.cpubenchmark.net.

[38] R. Gonzalez and M. Horowitz, "Energy dissipation in general purpose microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 9, pp. 1277–1284, 1996.

[39] J. A. Rice, *Mathematical Statistics and Data Analysis*. Duxbury Press, 2007.

[40] American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc. (ASHRAE), "Thermal Guidelines for Data Processing Environments - Expanded Data Center Classes and Usage Guidance," *ASHRAE Technical Committee 9.9 Whitepaper*, 2011.

[41] Q. Tang, T. Mukherjee, S. Gupta, and P. Cayton, "Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters," in *Proceedings of the Fourth International Conference on Intelligent Sensing and Information Processing*, ICISIP '06, pp. 203–208, 2006.

[42] M. Kesavan, I. Ahmad, O. Krieger, R. Soundararajan, A. Gavrilovska, and K. Schwan, "Practical compute capacity management for virtualized datacenters," *IEEE Transactions on Cloud Computing*, 2013.

[43] I. Paul, S. Yalamanchili, and L. K. John, "Performance impact of virtual machine placement in a datacenter," in *IEEE 31st International Performance Computing and Communications Conference*, IPCCC, pp. 424–431, 2012.

[44] N. Odeh, T. Grassie, D. Henderson, and T. Muneer, "Modelling of flow rate in a photovoltaic-driven roof slate-based solar ventilation air preheating system," *Energy conversion and management*, vol. 47, no. 7, pp. 909–925, 2006.

[45] iMPACT Lab, "BlueTool." http://impact.asu.edu/BlueTool.

[46] "Parallel Workloads Archive: The Standard Workload Format." http://www.cs.huji.ac.il/labs/parallel/workload/swf.html.

[47] J. Mars, L. Tang, R. Hundt, K. Skadron, and M. L. Soffa, "Bubble-up: Increasing utilization in modern warehouse scale computers via sensible co-locations," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, pp. 248–259, 2011.

[48] D. Meisner, C. Sadler, L. Barroso, W. Weber, and T. Wenisch, "Power management of online data-intensive services," in *Proceedings of the 38th annual international symposium on Computer Architecture*, ISCA-38, pp. 319–330, 2011.

[49] C. Bash, C. Patel, and R. Sharma, "Dynamic thermal management of air cooled data centers," in *Proceedings of the Tenth Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronics Systems*, ITHERM '06, pp. 445–452, 2006.

[50] C. Patel, R. Sharma, C. Bash, and A. Beitelmal, "Thermal considerations in cooling large scale high compute density data centers," in *Proceedings of the Eighth Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, ITHERM '02, pp. 767–776, 2002.

[51] F. Ahmad and T. Vijaykumar, "Joint optimization of idle and cooling power in data centers while maintaining response time," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS, 2010.

[52] J. Hamilton, "Where does the power go in high-scale data centers (keynote address)," in *Proceedings of the International Conference on Measurement and Modeling of Computer Systems*, Sigmetrics '09, 2009.

[53] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proceedings of the 34th annual International Symposium on Computer Architecture*, (New York), pp. 13–23, 2007.

[54] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, "Full-system Power Analysis and Modeling for Server Environments," in *Workshop on Modeling, Benchmarking, and Simulation (MoBS)*, 2006.

[55] S. Yeo, M. M. Hossain, J.-C. Huang, and H.-H. S. Lee, "Atac: Ambient temperature-aware capping for power efficient datacenters," in *Proceedings of the 5th annual Symposium on Cloud Computing*, SoCC, 2014.

[56] S. Yeo and H.-H. S. Lee, "Simware: A holistic warehouse-scale computer simulator," *Computer*, vol. 45, no. 9, pp. 48–55, 2012.

[57] A. Banerjee, T. Mukherjee, G. Varsamopoulos, and S. K. S. Gupta, "Cooling-aware and thermal-aware workload placement for green hpc data centers," *International Conference on Green Computing*, pp. 245–256, 2010.

[58] S. Biswas, M. Tiwari, T. Sherwood, L. Theogarajan, and F. Chong, "Fighting fire with fire: modeling the datacenter-scale effects of targeted superlattice thermal management," in *Proceeding of the 38th annual international symposium on Computer architecture*, ISCA-38, pp. 331–340, 2011.

[59] M. Patterson, "The effect of data center temperature on energy efficiency," in *11th Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems, ITHERM '08*, pp. 1167–1174, 2008.

[60] F. Pollack, "New microarchitecture challenges in the coming generations of cmos process technologies (keynote address)," in *Proceedings of the 32nd annual international symposium on Microarchitecture*, MICRO-32, p. 2, 1999.

[61] D. H. Woo and H.-H. S. Lee, "Extending amdahl's law for energy-efficient computing in the many-core era," *Computer*, vol. 41, no. 12, pp. 24–31, 2008.

[62] T. Mukherjee, A. Banerjee, G. Varsamopoulos, S. Gupta, and S. Rungta, "Spatio-temporal thermal-aware job scheduling to minimize energy consumption in virtualized heterogeneous data centers," *Computer Networks*, vol. 53, no. 17, pp. 2888–2904, 2009.

[63] J. Wilkes, "More Google cluster data." Google research blog, Nov. 2011. Posted at http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.

[64] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format + schema," technical report, Google Inc., Mountain View, CA, USA, Nov. 2011. Revised 2012.03.20. Posted at http://code.google.com/p/googleclusterdata/wiki/TraceVersion2.

[65] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Towards understanding heterogeneous clouds at scale: Google trace analysis," Tech. Rep. ISTC–CC–TR–12–101, Intel science and technology center for cloud computing, Carnegie Mellon University, Pittsburgh, PA, USA, Apr. 2012. Posted at http://www.istc-cc.cmu.edu/publications/papers/2012/ISTC-CC-TR-12-101.pdf.

[66] Intel, "Your Source for Information on Intel Products." http://ark.intel.com.

[67] D. Wiles, "Details of Intel Xeon Phi coprocessors." http://www.cpu-world.com/news_2012/2012080201_Details_of_Intel_Xeon_Phi_ coprocessors.html.

[68] D. Atwood and J. Miner, "Reducing data center cost with an air economizer," *Intel White Paper, Tech. Rep*, 2008.

[69] X. Wang and M. Chen, "Cluster-level feedback power control for performance optimization," in *Proceedings of the 14th International Symposium on High Performance Computer Architecture*, HPCA-14, pp. 101–110, 2008.

[70] D. Li, B. R. D. Supinski, M. Schulz, K. W. Cameron, and D. S. Nikolopoulos, "Hybrid mpi/openmp power-aware computing," in *Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium*, IPDPS, 2010.

[71] M. E. Haque, K. Le, Í. Goiri, R. Bianchini, and T. D. Nguyen, "Providing green slas in high performance computing clouds," in *International Green Computing Conference*, IGCC, 2013.

[72] S. Govindan, A. Sivasubramaniam, and B. Urgaonkar, "Benefits and limitations of tapping into stored energy for datacenters," in *Proceeding of the 38th annual international symposium on Computer architecture*, ISCA-38, pp. 341–352, 2011.

[73] H. Endo, H. Kodama, H. Fukuda, T. Sugimoto, T. Horie, and M. Kondo, "Effect of climatic conditions on energy consumption in direct fresh-air container data centers," in *International Green Computing Conference*, IGCC, 2013.

[74] C. Li, A. Qouneh, and T. Li, "iswitch: coordinating and optimizing renewable energy powered server clusters," in *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ISCA-39, 2012.

[75] I. n. Goiri, W. Katsak, K. Le, T. D. Nguyen, and R. Bianchini, "Parasol and greenswitch: Managing datacenters powered by renewable energy," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS, 2013.

[76] Í. Goiri, W. Katsak, K. Le, T. D. Nguyen, and R. Bianchini, "Designing and managing datacenters powered by renewable energy," *IEEE Micro*, 2014.

[77] N. Tolia, Z. Wang, P. Ranganathan, C. Bash, M. Marwah, and X. Zhu, "Unified thermal and power management in server enclosures," in *the ASME/Pacific Rim Technical Conference and Exhibition on Packaging and Integration of Electronic and Photonic Systems, MEMS, and NEMS*, InterPACK, 2009.

[78] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. H. Loh, D. McCauley, P. Morrow, D. W. Nelson, D. Pantuso, *et al.*, "Die stacking (3d) microarchitecture," in *Proceedings of the 39th Annual International Symposium on Microarchitecture*, MICRO-39, 2006.

[79] J. Sim, G. H. Loh, H. Kim, M. O'Connor, and M. Thottethodi, "A mostly-clean dram cache for effective hit speculation and self-balancing dispatch," in *Proceedings of the 2012 45th Annual International Symposium on Microarchitecture*, MICRO-45, 2012.

[80] J. Sim, G. H. Loh, V. Sridharan, and M. O'Connor, "Resilient die-stacked dram caches," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ISCA-40, 2013.

[81] I. Paul, V. Ravi, S. Manne, M. Arora, and S. Yalamanchili, "Coordinated energy management in heterogeneous processors," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC-13, 2013.

[82] K. Natarajan, H. Hanson, S. Keckler, C. Moore, and D. Burger, "Microprocessor Pipeline Energy Analysis," in *Proceedings of the 2003 International Symposium on Low Power Electronics and Design*, pp. 282–287, 2003.

[83] Anand Lal Shimpi, "Intel's Atom Architecture: The Journey Begins." http://www.anandtech.com/show/2493, 2008.

[84] S. Li, J. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 469–480, 2009.

[85] Semiconductor Industries Association, "Model for Assessment of CMOS Technologies and Roadmaps (MASTAR)." http://www.itrs.net/models.html, 2007.

[86] C. Auth, A. Cappellani, J. Chun, A. Dalis, A. Davis, T. Ghani, G. Glass, T. Glassman, M. Harper, M. Hattendorf, *et al.*, "45nm High-k+ metal gate strain-enhanced transistors," in *2008 Symposium on VLSI Technology*, pp. 128–129, 2008.

[87] S. Narendra and A. Chandrakasan, *Leakage in nanometer CMOS technologies*. Springer-Verlag New York Inc, 2006.

[88] M. Qureshi, M. Franceschini, and L. Lastras-Montano, "Improving read performance of phase change memories via write cancellation and write pausing," in *Proceedings of the International Symposium on High Performance Computer Architecture*, 2010.

[89] D. Ielmini, A. Lacaita, and D. Mantegazza, "Recovery and drift dynamics of resistance and threshold voltages in phase-change memories," *IEEE Transactions on Electron Devices*, vol. 54, no. 2, pp. 308–315, 2007.

[90] D. Ielmini, S. Lavizzari, D. Sharma, and A. Lacaita, "Physical interpretation, modeling and impact on phase change memory (PCM) reliability of resistance drift due to chalcogenide structural relaxation," in *Proceedings of the IEEE International Electron Devices Meeting*, IEDM, pp. 939–942, 2007.

[91] T. Nirschl, J. Phipp, T. Happ, G. Burr, B. Rajendran, M. Lee, A. Schrott, M. Yang, M. Breitwisch, C. Chen, *et al.*, "Write strategies for 2 and 4-bit multi-level phase-change memory," in *Proceedings of the IEEE International Electron Devices Meeting*, IEDM, pp. 461–464, 2007.

[92] M. Awasthi, M. Shevgoor, K. Sudan, B. Rajendran, R. Balasubramonian, and V. Srinivasan, "Efficient Scrub Mechanisms for Error-Prone Emerging Memories," in *Proceedings of the International Symposium on High Performance Computer Architecture*, HPCA, 2012.

[93] W. Xu and T. Zhang, "A time-aware fault tolerance scheme to improve reliability of multilevel phase-change memory in the presence of significant resistance drift," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 8, pp. 1357–1367, 2011.

[94] W. Zhang and T. Li, "Helmet: A resistance drift resilient architecture for multi-level cell phase change memory system," in *Proceedings of 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*, pp. 197–208, 2011.

[95] N. Papandreou, H. Pozidis, T. Mittelholzer, G. Close, M. Breitwisch, C. Lam, and E. Eleftheriou, "Drift-tolerant multilevel phase-change memory," in *2011 3rd IEEE International Memory Workshop (IMW)*, pp. 1–4, IEEE.

[96] Y. Hwang, C. Um, J. Lee, C. Wei, H. Oh, G. Jeong, H. Jeong, C. Kim, and C. Chung, "MLC PRAM with SLC write-speed and robust read scheme," in *Proceedings of the 2010 Symposium on VLSI Technology (VLSIT)*, pp. 201–202, 2010.

[97] B. Schroeder, E. Pinheiro, and W. Weber, "Dram errors in the wild: a large-scale field study," in *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, pp. 193–204, ACM, 2009.

[98] N. H. Seong, D. H. Woo, V. Srinivasan, J. A. Rivers, and H.-H. S. Lee, "SAFER: Stuck-at-fault error recovery for memories," in *Proceedings of the 43rd IEEE/ACM International Symposium on Microarchitecture*, 2010.

[99] N. H. Seong, D. H. Woo, and H.-H. S. Lee, "Security Refresh: Protecting Phase-Change Memory against Malicious Wear Out," *IEEE Micro*, vol. 31, no. 1, pp. 119–127, 2011.

[100] Y. Choi, I. Song, M.-H. Park, H. Chung, S. Chang, B. Cho, J. Kim, Y. Oh, D. Kwon, J. Sunwoo, J. Shin, Y. Rho, C. Lee, M. G. Kang, J. Lee, Y. Kwon, S. Kim, J. Kim, Y.-J. Lee, Q. Wang, S. Cha, S. Ahn, H. Horii, J. Lee, K. Kim, H. Joo, K. Lee, Y.-T. Lee, J. Yoo, and G. Jeong, "A 20nm 1.8V 8Gb PRAM with 40MB/s Program Bandwidth," in *Technical Digest of the 2012 IEEE International Solid-State Circuits Conference*, 2012.

[101] R. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[102] R. Bose and D. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and control*, vol. 3, no. 1, pp. 68–79, 1960.

[103] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres*, vol. 2, no. 2, pp. 147–156, 1959.

[104] R. S. Amant, A. Yazdanbakhsh, J. Park, B. Thwaites, H. Esmaeilzadeh, A. Hassibi, L. Ceze, and D. Burger, "General-purpose code acceleration with limited-precision analog computation," in *Proceedings of the 41st Annual International Symposium on Computer Architecture*, ISCA, 2014.

[105] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," in *Proceedings of the International Symposium on Microarchitecture*, MICRO, 2012.

[106] A. Sampson, J. Nelson, K. Strauss, and L. Ceze, "Approximate storage in solid-state memories," in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 25–36, 2013.

[107] W. Baek and T. M. Chilimbi, "Green: a framework for supporting energy-conscious programming using controlled approximation," in *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, vol. 45, pp. 198–209, 2010.

[108] S. Sidiroglou-Douskos, S. Misailovic, H. Hoffmann, and M. Rinard, "Managing performance vs. accuracy trade-offs with loop perforation," in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pp. 124–134, 2011.

[109] L. N. Chakrapani, B. E. Akgul, S. Cheemalavagu, P. Korkmaz, K. V. Palem, and B. Seshasayee, "Ultra-efficient (embedded) soc architectures based on probabilistic cmos (pcmos) technology," in *Proceedings of the conference on Design, automation and test in Europe*, pp. 1110–1115, European Design and Automation Association, 2006.

[110] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn, "Flikker: saving dram refresh-power through critical data partitioning," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS, 2011.

[111] F. Bedeschi, R. Fackenthal, C. Resta, E. M. Donze, M. Jagasivamani, E. C. Buda, F. Pellizzer, D. W. Chow, A. Cabrini, G. Calvi, *et al.*, "A bipolar-selected phase change memory featuring multi-level cell storage," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 217–227, 2009.

[112] D. Kang, J. Lee, J. Kong, D. Ha, J. Yu, C. Um, J. Park, F. Yeung, J. Kim, W. Park, *et al.*, "Two-bit cell operation in diode-switch phase change memory cells with 90nm technology," in *Proceedings of 2008 Symposium on VLSI Technology*, pp. 98–99, 2008.

[113] S. Yeo, N. H. Seong, and H.-H. S. Lee, "Can multi-level cell pcm be reliable and usable? analyzing the impact of resistance drift," in *Workshop on Duplicating, Deconstructing and Debunking*, 2012.

[114] L. Jiang, B. Zhao, Y. Zhang, J. Yang, and B. R. Childers, "Improving write operations in mlc phase change memory," in *Proceedings of the International Symposium on High Performance Computer Architecture*, HPCA, 2012.

[115] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman, "Enerj: Approximate data types for safe and general low-power computation," in *32nd ACM SIGPLAN conference on Programming Language Design and Implementation (PLDI)*, vol. 46, pp. 164–174, ACM, 2011.

[116] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Architecture support for disciplined approximate programming," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS, 2012.

[117] S. Narayanan, J. Sartori, R. Kumar, and D. L. Jones, "Scalable stochastic processors," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 335–338, European Design and Automation Association, 2010.

[118] H. Hoffmann, S. Sidiroglou, M. Carbin, S. Misailovic, A. Agarwal, and M. Rinard, "Dynamic knobs for responsive power-aware computing," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS, 2011.

[119] M. de Kruijf, S. Nomura, and K. Sankaralingam, "Relax: An architectural framework for software recovery of hardware faults," in *Proceedings of the International Symposium on Computer Architecture*, ISCA, 2010.

[120] B. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative," in *Proceedings of the International Symposium on Computer Architecture*, ISCA, 2009.

[121] M. Qureshi, V. Srinivasan, and J. Rivers, "Scalable high performance main memory system using phase-change memory technology," in *Proceedings of the 36th International Symposium on Computer Architecture*, ISCA, 2009.

[122] M. K. Qureshi, M. M. Franceschini, L. A. Lastras-Montaño, and J. P. Karidis, "Morphable memory system: a robust architecture for exploiting multi-level phase change memories," in *Proceedings of the 37st Annual International Symposium on Computer Architecture*, ISCA, 2010.

[123] M. K. Qureshi, J. Karidis, M. Fraceschini, V. Srinivasan, L. Lastras, and B. Abali, "Enhancing Lifetime and Security of Phase Change Memories via Start-Gap Wear Leveling," in *Proceedings of the International Symposium on Microarchitecture*, MICRO, 2009.

[124] M. Wuttig, "Phase-change materials: Towards a universal memory?," *Nature materials*, vol. 4, no. 4, pp. 265–266, 2005.

[125] N. H. Seong, S. Yeo, and H.-H. S. Lee, "Tri-level-cell phase change memory: Toward an efficient and reliable memory system," in *Proceedings of the 40st Annual International Symposium on Computer Architecture*, ISCA, pp. 440–451, 2013.