Stony Brook University

GreenDM: A Versatile Tiering Hybrid Drive for the Trade-Off Evaluation of Performance, Energy, and Endurance

A Dissertation Presented

by

Zhichao Li

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

May 2014

Stony Brook University

The Graduate School

Zhichao Li

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree, hereby recommend
acceptance of this dissertation.

Erez Zadok–Dissertation Advisor
Associate Professor, Computer Science Department

Scott D. Stoller–Chairperson of Defense
Professor, Computer Science Department

Michael Ferdman–Third Inside Member
Assistant Professor, Computer Science Department

Dr. Andrew W. Leung–Outside Member
Software Architect, Computer Science, Formation Data Systems

This dissertation is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

GreenDM: A Versatile Tiering Hybrid Drive for the Trade-Off Evaluation of Performance, Energy, and Endurance

by

Zhichao Li

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

2014

There are trade-offs among performance, energy, and device endurance for storage systems. These trade-offs become more complex in storage system comprising different storage technologies. Designs optimized for one dimension or workload often suffer in another. Therefore, it is important to study the trade-offs so as to be able to adapt the system to workloads. As different types of drives have different traits, tiering hybrid drives are studied more closely. However, previous tiering hybrids are often designed for high throughput, efficient energy consumption, or improving endurance—leaving empirical study on the trade-offs being unexplored. Past endurance studies also lack a concrete model and metric to help study the trade-offs. Lastly, previous designs are often based on inflexible policies that cannot adapt easily to changing conditions.

We designed and developed *GreenDM*, a versatile tiering hybrid drive that combines Flash-based SSDs with traditional HDDs; we present our endurance model to study the aforementioned trade-offs. GreenDM presents a block interface and requires no modifications to existing application software. GreenDM migrates hot data to the faster SSD and cold data to the slower HDD. GreenDM offers tunable parameters useful in adapting the system to many workloads. We have designed, developed, and carefully evaluated GreenDM with a variety of workloads using commodity SSD and HDD drives. We demonstrated the importance of versatility to be able to adapt to various workloads.

Our thesis is that one must study the trade-offs among performance, energy, and endurance, especially in the ever more popular tiered storage systems, to enable adaptation to workloads. Our system is versatile so that it can adapt to different workloads to achieve certain trade-offs by adjusting the important system parameters. We also provide several interesting observations along the cost dimension. We developed a cost model for GreenDM and evaluated it under realistic cost metrics. Future storage system designs have to consider multiple optimizations dimensions: performance, energy, endurance, and dollar cost.

We close with several interesting long-term future research. First, it will be interesting to provide automated control knobs for users to trade-off performance, energy efficiency, and endurance. Second, one could extend the two-tier system to three tiers and explore more tiering policies. Third, it would be useful to add security as an additional dimension to further explore these trade-offs. Forth, one could experiment with different storage devices and policies in the future, and help build more efficient storage systems to achieve high performance at minimum cost. Fifth and last, it would be interesting to provide control support at the CPU level as well to further justify the trade-offs among performance, energy, and endurance.

To my family that supports me from a far.

Contents

List of Figures

List of Tables

Acknowledgments

Doing Ph.D. in Computer Science is not an easy task, especially for international students. There are many challenging issues for them to solve carefully: (1) their parents are getting older and may be hospitalized during the long period, but it is extremely difficult to provide financial and emotional support from far while doing a Ph.D. in U.S.; (2) the intensive working pressure from doing the Ph.D. subject and the emotional pressure of being abroad can potentially affect the student's health condition in a negative way, if the pressures are not taken care of; and (3) they may question themselves many times as for where the destination is. To give it up or carry it on, to turn left or right, those are indeed decisions they will have to make to survive.

As such an international student, my experience is filled with challenges. I feel lucky to eventually enter this stage after walking through so many tough situations. In for a penny, in for a pound. That is the reason.

Other than the tough situations, there are also joys that come from the fulfillment of the work. This work is made possible with enormous help from many people across several countries. I really appreciate people's help and as a return, I set "contributing to the community" as one of my life purposes.

Professor Erez Zadok advised me for the whole Ph.D. journey with patience and encouragement. He had my foot in the door of advanced research, pointed me in the right direction, and funded me for my whole Ph.D. research. Without him, my research would end nowhere. More importantly, Professor Zadok has tolerated and corrected hundreds of mistakes I made over the years. It is those mistakes, other than the papers I published, that eventually make me a better researcher, a better developer, and a better person. I am proud that I am a "FSLer."

My family, back in China, has supported me with the greatest love of the world ever since I was born, and has encouraged me a lot when I was down. They do not know English, they could not use a computer, they do not know how research in Computer Science looks like, but they do know how to trust and love their son. It is my family that keeps me going like ever before.

My host family, Susan M. Colatosti and Carlo Colatosti, has been there supporting me for five years. It was a great pleasure to have dinner with them from time to time, discussing topics from Information Technology to Universe and from English learning to International jokes. We also had wonderful time together doing sightseeing. Those experiences did open my eyes from a different perspective.

Professors Radu Grosu, Scott D. Stoller, and Scott A. Smolka helped me with my early Ph.D. work. When I was theoretically challenged, they would stand there and get my feet off the ground.

Dr. Vasily Tarasov, who recently graduated from the FSL, is a good friend of mine. He would go the extra mile to provide helps whenever in need. More importantly, he encouraged me many times when I was down in research, and respected me a lot for my work. It would certainly be a dream ticket to work with him in the future because he is so helpful and encouraging, and he respects the other people and their work a lot.

Several students in the lab have helped my projects by working directly with me. For example, Koundinya Muppalla and Priya Sehgal helped me on the control theory work. Rajesh Aavuty and Ming Chen helped me on the "GreenDM" project. Amanpreet Mukker helped me on the expansion of the "GreenDM" project. Madhu Srikar Palmur helped me on the "CPUIDLE" project and eventually took it over from me. Many thanks to the people mentioned above. Without their

help, my research would be largely delayed.

Drs. Kevin M. Greenan and Andrew W. Leung provided me one summer internship in the Backup and Recovery Systems (BRS) division of EMC, and co-authored one USENIX FAST paper with me. Their help during the internship certainly influenced my research that followed. Not only have they taught me how to identify problems, but they also shared their domain knowledge with me. Many thanks also to my internship project manager David Brittle who taught me how to look at projects from a global perspective.

Chirag Bhatt, Aalap Desai, and Rajit Kambo provided me one another summer internship in VMware, and co-authored with me one paper in the Feedback Computing Workshop. Many thanks to Dr. Xiaoyun Zhu for her valuable advice on the internship project. That internship had great impact on my research that followed because it opened a new direction for my research. Moreover, I also realized the importance of having patience to do research during the internship.

Dan Green, Mayank Rawat, and Sonic Wang offered me a second summer internship in VMware, which was eventually transferred into a full-time offer as a software engineer. I learned the most from this internship because the subject on distributed platform storage was what I had always been crazy about but never got a chance to get my hands dirty on. Many thanks to this memorable experience, I am more certain about my future career pursue.

Drs. Justin Seyster and Richard P. Spillane, who graduated recently from FSL, have helped me drawing the picture of a Ph.D. journey, and advised me on how to avoid the hidden pitfalls. Moreover, all other students in the FSL that I know have been a good company. I still remember the joyful time when we played together during the research break.

In addition, I would like to thank several close friends of mine across countries for their technical and emotional support so that I do not feel alone on the road. They are Peng Chen, Shimin Gong, Hao Huang, Xing Lin, Xiaojun Liu, Yanhua Li, Lei Xiao, Qian Zhao, Jiangfu Zhou, and Heng Zhao. Many thanks to Drs. Dan Feng and Zhan Shi because they provided me generous resources and instructions during my undergraduate research years. I was taking such a great shower of research at those early years so that I kept my heart set on for a Ph.D. degree since then.

Finally, I would like to also thank my committee members, Professors Michael Ferdman and Scott D. Stoller, Dr. Andrew W. Leung—and again my adviser—Professor Erez Zadok, for their volunteer work to serve as my thesis committee. I do appreciate that.

Chapter 1

Introduction

The total amount of electronic data stored world-wide is rising exponentially. By 2020, that figure is expected to reach 35 Zetta Bytes [41]. It challenges how fast the storage systems can be to store and fetch the data. Studies show that power consumption in the IT infrastructure is critical [22, 71, 77], up to 40% power consumption of which comes from storage [118]. Therefore, power consumption has become an important factor influencing storage systems design [5, 79, 91, 119, 122, 132, 139, 153]. Modern computer components such as CPU, RAM, and disk drives tend to have multiple power states with different operational modes [29,50,79]. Among them, traditional magnetic HDDs achieve the worst *power-proportionality* [10], which states that systems should consume power proportional to the amount of work performed. Moreover, as failures in storage systems become a serious concern [7, 21, 32, 54, 63, 90, 102, 114, 148], the endurance of storage devices matters as well.

Different storage devices differ in speed, capacity, cost, endurance, and power consumption [104]. There are trade-offs among these dimensions in storage systems combining different types of devices. Designs optimized for one dimension or workload often suffer in other dimensions and workloads. Moreover, in prior work, we analyzed the energy and performance profiles of server workloads, such as Web servers, email servers, database servers, and file compression [72, 79, 119]. We discovered large deviations for both performance and energy consumption—as much as 10 times—suggesting that there are significant opportunities to save energy and improve performance. Therefore, it is important to study the trade-offs among these dimensions, and develop highly versatile solution to enable adaptation to different workloads.

With the advent of Flash-based Solid State Drives (SSDs) that are more power and performance efficient than HDDs, many considered SSDs as the front tier storage cache (e.g., EMC's FAST [73], VMware's vFlash [133], IBM's Tiering system [4], etc). While it is beneficial to explore SSDs as cache, there is a trend in industry to come up with hybrid drive exploring SSDs as the primary storage to achieve better trade-offs among performance, cost, and capacity. Examples are: (1) Apple's Fusion Drive [141]; (2) Microsoft's Ready Drive [100]; (3) Western Digital's Solid State Hybrid Drive (SSHD) [138]; (4) Nimble's CASL [94]; (5) Tintri's VMstore [134]; and (6) Dell even sells a Compellent Flash Array [30] that combine two types of SSDs together—Single-Level Cell (SLC) and Multi-Level Cell (MLC)—to achieve the above trade-offs.

Many such approaches often aimed for high performance [23, 49, 68, 123, 129, 147], efficient energy consumption [49, 147], or improved endurance [68, 147]. Therefore, study on the trade-offs among performance, energy, and endurance is largely unexplored. Studying the trade-offs can

help understand the relationship among performance, energy, and endurance, and can also help build storage systems that can adapt to different workloads and dimensions. Moreover, current endurance studies are missing a concrete endurance model and metric to help explore the above trade-offs. In addition, past studies often had designs with fixed or inflexible policies that made it difficult to adapt to different workloads. Moreover, many previous approaches usually rely on simulations and refer to manufacturer's energy and performance specifications for benchmarks, instead of using empirical, real-world experiments.

We designed and implemented a Linux Device Mapper [140] (DM) target named *GreenDM*, and came up with a concrete endurance model and metric to study the trade-offs. GreenDM receives data requests from the tiering hybrid virtual device, and then transparently redirects the resulting requests to the underlying block devices. The DM framework offers additional benefits: it can be used with any target device (e.g., replication, multi-path, encryption, redundancy, and snapshots). The DM framework is also highly scalable: one can easily configure the virtual device to use multiple physical devices transparently.

GreenDM separates hot data from cold data based on their access patterns: hot data is stored on the SSD and cold data is stored on the HDD. When cold data becomes hot, GreenDM migrates it from the HDD to the SSD; conversely, when hot data becomes cold or more space is needed for hotter data, GreenDM migrates colder data from the SSD to the HDD.

By utilizing the SSD for hot data before using the HDD, GreenDM improves performance and reduces energy use—as SSDs are typically faster and consume less energy than HDDs. To improve concurrency, GreenDM decouples the migrations between the SSD and the HDD. By keeping mostly cold data on the HDD, GreenDM can spin down the HDD at times and help reduce whole-system energy consumption. By counting the number of physical SSD reads and writes [84, 97] and the HDD start-stop cycles [47], GreenDM tracks vital parameters that impact the endurance of the underlying storage devices.

To enable adaptation to different workloads, GreenDM supports several versatile configuration parameters to determine the migration thresholds between the SSD and the HDD. These parameters can be tuned in accordance with the workloads.

We have evaluated GreenDM with several workloads. We experimented with several configurable GreenDM parameters, analyzed the results, and demonstrated their impact on the trade-offs among performance, energy, and endurance. We also showed the importance of matching configuration parameters to specific workloads to tune the above trade-offs. In the FIU online trace workload, for example, we showed that various GreenDM configurations achieved a higher throughput (58–142%) than Mylinear (a simple hybrid without any additional data management), but consumed more power (4–8%) and further reduced the SSD's endurance by 11–15% . A larger extent size (ES) lead to higher throughput and larger energy savings, but reduced the SSD's endurance further.

Our thesis is that one must study the trade-offs among performance, energy, and endurance, especially in the ever more popular tiered storage systems, to enable adaptation to workloads and dimensions since there is no one-to-all solution. Our system is versatile so that it can adapt to different workloads to achieve certain trade-offs by adjusting the important system parameters. We also provide several interesting observations along the cost dimension. We developed a cost model for GreenDM and evaluated it under realistic cost metrics. Future storage system designs have to consider multiple optimizations dimensions: performance, energy, endurance, and dollar cost.

We further bring up several interesting long-term future research directions. First, it will be

interesting to provide automated control knobs for the user to trade-off performance, energy efficiency, and endurance. Second, one could extend the two-tier system to three tiers and explore more tiering policies. Third, it would be useful to add security as a forth dimension to further explore the trade-offs. Forth, GreenDM is transparent and flexible so that we can easily experiment with different storage devices and policies in the future, and help build more efficient storage systems ready for the emerging trend and trade-offs of various storage devices — Flash, Phase Change Memories (PCMs), and Shingled Magnetic Recording (SMR) drives to achieve the high performance efficiency with the minimum cost. Last but not least, it would be interesting to provide control support at the CPU level as well to further justify the trade-offs among performance, energy, and endurance.

The rest of the thesis is organized as follows: Chapter 2 shares some of the background knowledge that can help the reader better understand the rest of the thesis. Chapter 3 shows the lessons we learned along the way toward this thesis. Chapter 4 presents the work on power consumption in enterprise-scale backup storage systems. Chapter 5 presents the work on the energy consumption and performance of systems software. Chapter 6 introduces the work on a versatile hybrid drive for the trade-offs evaluation among performance, energy, and endurance. Chapter 7 presents interesting observations regarding the associated cost dimension of GreenDM. Chapter 8 presents follow-up work on our caching system based on the current hardware and software setup. Chapter 9 presents follow-up work on evaluating both the tiering and caching systems with a different capacity ratio of SSD over total. Chapter 10 lists several future work that go beyond this dissertation. Chapter 11 concludes this thesis.

Chapter 2

Background

In this chapter, we share some background knowledge on: (1) the trade-offs among performance, energy, power, and endurance in storage systems; (2) the justification of choosing tiering over caching; and (3) the endurance study. The knowledge is helpful to better illustrate the work in the thesis.

2.1 Trade-Offs

Trade-offs are everywhere in systems. There are trade-offs among performance, energy, power, endurance, cost, and capacity in storage systems. We show some examples to help illustrate the topics.

In terms of the trade-offs between performance and capacity, let us take the personal computer for example. Suppose the budget to buy a personal computer is fixed. If one wants to have high I/O throughput, then the SSD is a good option. However, the storage capacity is going to be smaller compared with buying an HDD. The reason is that SSD is more expensive per gigabyte.

In terms of the trade-offs between performance and power, one interesting example is DVFS [156]. In a CPU-bound system, if the CPU frequency is higher, the CPU performance will be higher. A better CPU performance can largely lead to higher storage throughput. However, since the CPU is now running at a faster frequency, it consumes more power than when it runs at a lower frequency.

In terms of the trade-offs between the energy saving and the endurance concern [47], let us take the HDD for example. Spinning down the HDD reduces the energy consumption, but it can wear-out the HDD more since the HDD can only endure a limited number of start-stop cycles.

In terms of the trade-offs between endurance and performance, let us take ECC for example. Once a bit corrupts, the lost data can be corrected by Error-correcting Code. It makes the system more durable. However, it takes time to check the ECC and to resolve the bit error. Therefore, the overall performance of storage systems may be decreased accordingly.

Therefore, data management techniques have to consider possible trade-offs to fully explore the multi-dimensions of storage systems.

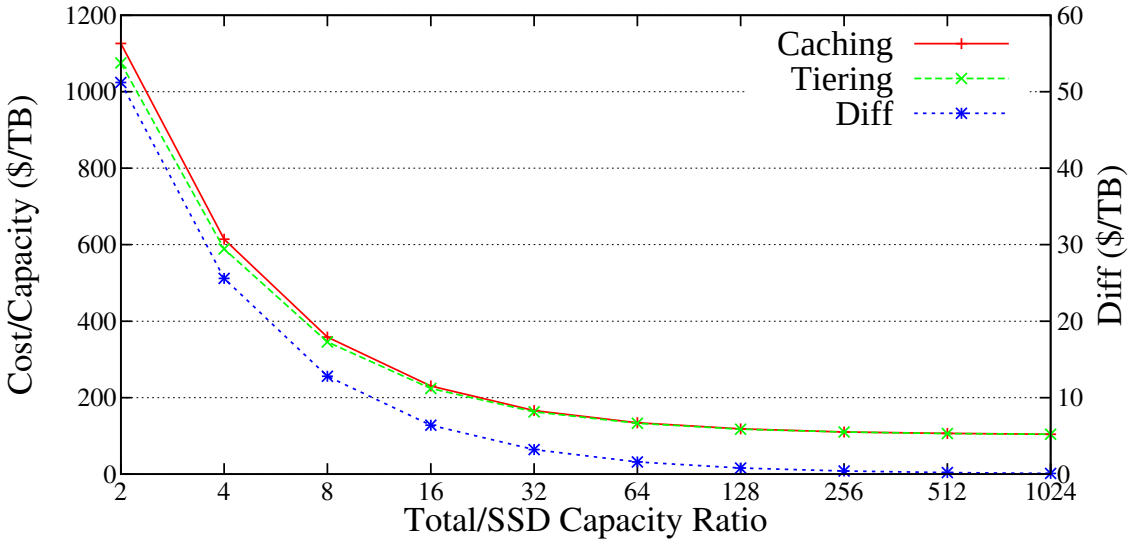


Figure 2.1: Tiering v.s. Caching hybrid justification. *Suppose we are to build 1TB drive. The normalized capacity cost of devices are based on Freitas's study [104]. Lower purchase cost over capacity is better. The x-axis is the total capacity over SSD capacity ratio. The "Cost/Capacity" y-axis is for the absolute caching and tiering results. The "Diff" y-axis is for the difference results of caching minus tiering.*

2.2 Tiering v.s. Caching Hybrid Justification

Hybrid drive storage systems can be either based on caching or tiering. Their ideas are very similar: frequently accessed data goes to faster device and less frequently accessed data goes to slower device. A tiering-based hybrid has the benefit of building a larger device capacity as there are no duplicate data copies across the devices. In the thesis, we choose tiering over caching because tiering can achieve much better trade-offs among throughput, capacity, and purchase cost when the device capacities in different tiers are similar and when the total device capacity is not too large.

To further illustrate that, we came up with a hybrid justification as shown in Figure 2.1. As we can see from the figure, the tiering-based architecture achieves better purchase cost over capacity metrics compared with caching based approach. The lower the total capacity over SSD capacity ratio is, the larger the purchase cost over capacity difference is. More specifically, when the SSD capacity is half the total capacity, tiering based architecture reduces the purchase cost over capacity by as much as 51.2 dollars per TB (i.e., 5%). Since our device capacities in different tiers are not largely different, tiering based architecture is a better option for us.

In fact, the research scope of tiering v.s. caching systems could go beyond our above discussion. To provide even more interesting results, we present the design and implementation of our caching system, together with the evaluation results in Chapter 8. For the sake of this dissertation, we provide the hybrid justification in this section to illustrate why we choose a versatile tiering hybrid system over a caching system under our experiment setup to study the trade-offs among performance, energy, and endurance.

2.3 Endurance Study

We learned about the endurance topic in storage systems through the survey of many published papers (i.e., up to 40). We summarize them below. Note that our work is focusing on device (hardware) endurance, instead of data reliability that depends not only on the hardware endurance, but also depends on software techniques (e.g., data copy and erasure coding).

2.3.1 Summary

Endurance is one fundamental issue of the IT infrastructure. There are several types of system failures: software failures, hardware failures, network failures, failures due to operator error, and failures due to environmental problems (e.g., power outages [21]). Failure happens in almost any subcomponent of the computing and storage systems, and usually happens more often than what the hardware manufacturer specified. For example, there are failures in processors [26, 113, 116], memory and drives [7, 21, 32, 45, 54, 61, 63, 66, 82, 90, 102, 114, 115, 117, 125, 148], and even the physical interconnects and protocol stacks [63].

The endurance issue will become even more significant when the system scales. When the system scales, the system becomes more complex and more vulnerable to failures. Therefore, it is interesting to gain more knowledge on the endurance issue.

2.3.2 Hardware Failure Factors

One of the main reasons for the hardware failures is the aging effect. Once the device is getting old, it is more likely that an failure will happen.

For CPUs, temperature is one factor that greatly influences the chip lifetime [26]. Therefore, CPU power management techniques is explored to control the CPU operating temperature to increase the chip's lifetime. For DRAMs, the utilization, instead of the temperature, affects the DRAM errors more [117]. Thus, ECC is used to correct the soft errors, and hardware replacement is adopted once there are hard errors. For HDDs, several factors affect their lifetime, including sector error, reallocation count, off-line reallocation count, the count of suspected "on-probation" bad sectors, number of power up-down cycles, vibration, etc [102]. ECC and recovery techniques are generally utilized to cope with the errors. For Flash-based SSDs, they suffer from the endurance problem because Flash device requires one block erasure operation before the block can be written. There are different levels of failure modes [66, 84, 97]. In failure mode I, the ECC can successfully correct the device bit errors. It happens when the device is not erased beyond the specified maximum value. In failure mode II, the bit errors rate goes beyond the ECC's correction capability, but the device is still within the retention period. This can happen when the device is erased beyond the specified maximum value. In failure mode III, even the device goes beyond the retention period. In our thesis, we examine the SSD endurance to be within failure mode I. There are three levels of techniques explored to overcome the chip endurance issues. At the circuit board level, solutions such as ECC have been widely implemented. At the system or whole product level, additional endurance features are being incorporated, including wear-leveling, DRAM cache regimes and RAID scrubbing techniques. At the data center level, there are hardware and software utilities to allow Flash to function reliably with remote data replicas.

Studies [113, 115] also show that the time between failures at a individual node, as well as at an entire system, is fit well by a gamma [142] or Weibull [143] distribution with decreasing hazard rate, instead of previously assumed exponential distribution. Moreover, mean repair times vary widely across systems, ranging from 1 hour to more than one day, depending on the size of the system.

2.3.3 Models

Endurance and reliability models of different computer components are worth exploring for several reasons. First, it can take a long time to observe device failures. Therefore, with such models, researchers can estimate the devices' health status. Second, the models can be further used to evaluate the trade-offs between performance and cost efficiency. There are many such models being explored. For example, several models [54, 90, 125] regarding the disk endurance have been discussed over the years. They look into the disk Self-Monitoring, Analysis and Reporting Technology (S.M.A.R.T.), and explore with either Bayesian based approach, or machine learning based algorithm, or combination of Bayesian and Markov models. The issue is that the estimation accuracy is not very high. There is also one work [45] arguing that MTTDL is not a good reliability metric for storage system reliability, and proposing a new metric called NOrmalized Magnitude of Data Loss (NOMDL) to better evaluate the storage system reliability. Netapp recently released an online tool [106] to calculate the reliability of RAID6. Regarding the SSD endurance model, study on SSD endurance normally refers to the amount of data that can be written to an SSD during its lifetime [74]. There is one study [131] exploring hardware-specific endurance model for SSD. While it is useful in some cases, it requires hardware parameters (e.g., voltage, density, etc.) to estimate the endurance through simulation, and can be inconvenient for user-level endurance estimation in reality.

Although it is true that the models can be useful, it is also true that the model validation is a difficult process. On one hand, it takes a long time to actually verify the devices' status. On the other hand, there can be other factors that affect the device endurance, other than the factors explored in the models. Therefore, research exploring such models needs to be expanded and examined carefully.

Chapter 3

Lessons Learned

We now provide a brief overview of our past work that led us to the topics introduced in Chapter 1. Having them all together, we touched the surface of the essence about power, performance, energy, and endurance, and eventually came up with the work to study the trade-offs among performance, energy, and endurance of hybrid drive storage system. Understanding these past efforts can also help understand the evaluation analysis in Chapters 6, 7, 8, and 9.

3.1 Elements of Past Study

3.1.1 Power: Power Consumption in Enterprise-Scale Backup Storage Systems

Power consumption has become an important factor in modern storage system design. Power efficiency is particularly beneficial in disk-based backup systems that store mostly cold data, have significant idle periods, and must compete with the operational costs of tape-based backup. There are no prior published studies on power consumption in these systems, leaving researchers and practitioners to rely on existing assumptions. In this work we present the first analysis of power consumption in real-world, enterprise, disk-based backup storage systems. We uncovered several important observations, including some that challenge conventional wisdom. We discuss their impact on future power-efficient designs. We present the details of this study in Chapter 4.

3.1.2 Energy and Performance: On the Energy Consumption and Performance of Systems Software

Models of energy consumption and performance are necessary to understand and identify system behavior, prior to designing advanced controls that can balance out performance and energy use. This work considers the energy consumption and performance of servers running a relatively simple file-compression workload. We found that standard techniques for system identification do not produce acceptable models of energy consumption and performance, due to the intricate interplay between the discrete nature of software and the continuous nature of energy and performance. This motivated us to perform a detailed empirical study of the energy consumption and performance of this system with varying compression algorithms and compression levels, file types, persistent

storage media, CPU DVFS levels, and disk I/O schedulers. Our results identify and illustrate factors that complicate the system's energy consumption and performance, including nonlinearity, instability, and multi-dimensionality. Our results provide a basis for future work on modeling energy consumption and performance to support principled design of controllable energy-aware systems. We present the details of this study in Chapter 5.

3.2 Put Together

3.2.1 GreenDM: A Versatile Tiering Hybrid Drive for the Trade-Off Evaluation of Performance, Energy, and Endurance

Putting all the lessons we learned together–not only the background knowledge but also the previous work, we now present our core thesis work.

There are trade-offs among performance, energy, and device endurance for storage systems. Designs optimized for one dimension or workload often suffer in another. Therefore, it is important to study the trade-offs and adapt the system to workloads. As different types of drives have different traits, hybrid drives are studied more closely. However, previous hybrids are often designed for high throughput, efficient energy consumption, or improving endurance—leaving empirical study on the trade-offs being unexplored. Past endurance studies also lack a concrete model and metric to help study the trade-offs. Lastly, previous designs are often based on inflexible policies that cannot adapt easily to changing conditions.

We build *GreenDM*, a versatile hybrid drive that combines Flash-based SSDs with traditional HDDs, and present our endurance model, to study the above trade-offs. GreenDM presents a block interface and requires no modifications to existing software. GreenDM migrates hot data to the faster SSD and cold data to the slower HDD. GreenDM offers tunable parameters to adapt the system to many workloads. We have designed, developed, and carefully evaluated GreenDM with a variety of workloads using commodity SSD and HDD drives. We demonstrated the importance of versatility to adapt to various workloads. We present the details of this study in Chapter 6.

3.2.2 Cost Evaluation

Modern storage systems are becoming more complex, especially in combining different storage technologies with vastly different behaviors. Performance or throughput alone is not enough to characterize storage systems: energy efficiency, durability, and more are becoming equally important. We posit that one must evaluate storage systems from a dollar cost perspective as well as performance. We also believe that the cost should consider the workloads in use over the expected lifetime of the storage systems. We designed and developed a versatile hybrid storage system under Linux that combines HDD and SSD. Our system includes many tunable parameters to be able to trade-off performance, energy use, and durability. We built a cost model and evaluated our system under a variety of workloads and parameters, to illustrate the importance of cost evaluations of storage systems. We provide the details of this study in Chapter 7.

3.2.3 Caching Follow-Up

Our environment setup is not optimal for a caching system, but for a tiering system, since caching is often deployed in fairly large storage systems. However, we chose to develop and evaluate such a caching system as well–and compare it with the tiering under the same hardware and software setup—so we can provide fair evaluations of both techniques under identical conditions.

Tiering and caching based hybrid approaches share several design traits with each other (e.g., similar data management policies). But, they are not the same approaches. There are existing studies [24, 52] exploring the pros and cons of the tiering and caching based approaches. However, there is no current work that builds the two realistic systems with similar strategies, and empirically evaluates the two systems from the cost perspective under the same environment, when SSDs are deployed. We present the details of this study in Chapter 8.

3.2.4 Capacity Ratio Follow-Up

The capacity ratio of SSD over total capacity matters for our hybrid drive in terms of throughput, energy and power, device endurance reduction, and dollar cost. Initially, we explored using $1/4$ as the capacity ratio of SSD over total capacity. We then also explored $1/8$ as the capacity ratio of SSD over total capacity. We reran all experiments and analyzed the results. We present these results in Chapter 9.

3.3 Conclusion

We discussed important lessons learned from past studies in Section 3 and showed how we came up with the trade-offs study, the cost study, the caching follow-up work, and the capacity ratio impact work. Next, we are going to explore the details of the power consumption study in enterprise-scale backup storage systems in Chapter 4.

Chapter 4

Power Consumption in Enterprise-Scale Backup Storage Systems

Power has become an important design consideration for modern storage systems as data centers now account for close to 1.5% of the world's total energy consumption [71], with studies showing that up to 40% of that power comes from storage [118]. Power consumption is particularly important for disk-based backup systems because: (1) they contain large amounts of data, often storing several copies of data in higher storage tiers; (2) most of the data is cold, as backups are generally only accessed when there is a failure in a higher storage tier; (3) backup workloads are periodic, often leaving long idle periods that lend themselves to low power modes [135, 152]; and (4) they must compete with the operational costs of low power, tape-based backup systems.

Even though there has been a significant amount of work to improve power consumption in backup or archival storage systems [24,101,122], as well as in primary storage systems [5,139,153], there are no previously published studies of how these systems consume power in the real world. As a result, power management in backup storage systems is often based on assumptions and commonly held beliefs that may not hold true in practice. For example, prior power calculations have assumed that the only power needed for a drive is quoted in the vendor's specification sheet [24, 122, 144]. However, an infrastructure, including HBAs, enclosures, and fans, is required to support these drives; these draw a non-trivial amount of power, which grows proportionally with the number of drives in the system.

In this chapter, we present the first study of power consumption in real-world, large-scale, enterprise, disk-based backup storage systems. We measured systems representing several different generations of production hardware using various backup workloads and power management techniques. Some of our key observations include considerable power consumption variations across seemingly similar platforms, disk enclosures that require more power than the drives they house, and the need for many disks to be in a low-power mode before significant power can be saved. We discuss the impact of our observations and hope they can aid both the storage industry and research communities in future development of power management technologies.

The remainder of the chapter is structured as follows. Section 4.1 discusses related power management and analysis work. Section 4.2 describes the systems measured and our experimental setup. Section 4.3 presents the results of our analysis and a discussion of our key observations. We then conclude in Section 4.4.

4.1 Related Work

Empirical power consumption studies have guided the design of many systems outside of storage. Mobile phones and laptop power designs, which are both sensitive to battery lifetime, were influenced by several studies [19, 81, 108, 110]. In data centers, studies have focused on measuring CPU [89, 109], OS [12, 13, 46], and infrastructure power consumption [9] to give an overview of where power is going and the impact various techniques have, such as dynamic voltage and frequency scaling (DVFS). Recently, Sehgal et al. [119] measured how various file system configurations impact power consumption.

Existing storage system power management has largely focused on managing disk power consumption. Much of this existing work assumes that as storage systems scale their capacity—particularly backup and archival systems—the number of disks will increase to the point where disks are the dominant power consumers. As a result, most solutions try to keep as many drives powered-off as possible, spun-down, or spun at a lower RPM. For example, archival systems like MAID [24] and Pergamum [122] use data placement, scrubbing, and recovery techniques that enable many of the drives in the system to be in a low-power mode. Similarly, PARAID [139] allows transitioning between several different RAID layouts to trade-off energy, performance, and reliability. Hibernator [153] allows drives in a RAID array to operate at various RPMs, reducing power consumption while limiting the impact to performance. Write Off-Loading [91] redirects writes from low-power disks to available storage elsewhere, allowing disks to stay in a low-power mode longer.

Our goal is to provide power consumption measurements from real-world, enterprise-scale backup systems, to help guide designs of power-managed storage systems.

4.2 Methodology

We measured several real-world, enterprise-class backup storage systems. Each used a Network-Attached-Storage (NAS) architecture with a storage controller connected to multiple, external disk drive enclosures. Figure 4.1 shows the basic system architecture. Each storage controller exports to file-based interfaces to clients, such as NFS and CIFS—and backup-based interfaces, such as VTL and those of backup software (e.g., Symantec's OST [96]). Each storage controller performs inline data deduplication; typically these systems contain more CPUs and memory than other storage systems to perform chunking and to maintain a chunk index.

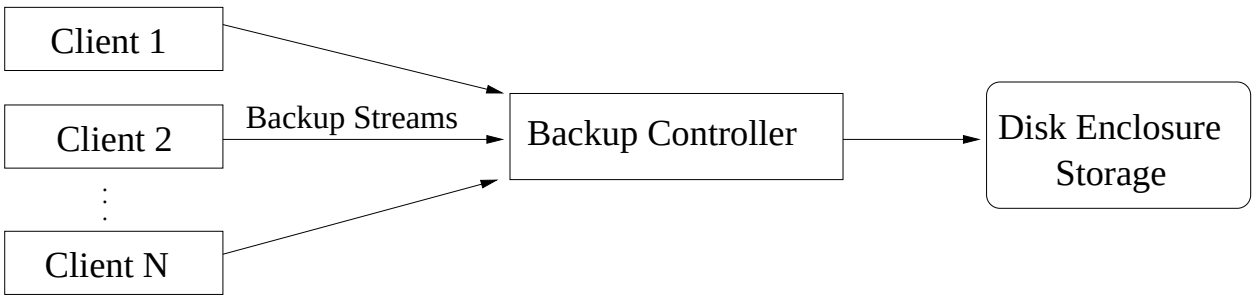


Figure 4.1: Backup system architecture

	DD880	DD670	DD860	DD990
Ship Year	2009	2010	2011	2012
Intel CPU	X7350	E5504	E5504	E7-4870
# CPUs	2	1	2	4
RAM	64GB	16GB	72GB	256GB
NVRAM	2GB	1GB	1GB	4GB
# Disks	4	7	4	4
# Pow Sup	2	2	2	4
# Fans	8	8	8	8
# NICs	1	1	1	2
# HBAs	3	1	3	4

Table 4.1: Controller hardware summary

Table 4.1 details the four different EMC controllers that we measured. Each controller was shipped or will be shipped in a different year and represents hardware upgrades over time. Each controller, except for DD670, stores all backup data on disks in external enclosures, and the four disks (three active plus a spare) in the controller store only system and configuration data. DD670 is a low-end, low-cost system that stores both user and system data in its seven disks (six active plus one spare). Each controller ran the same software version of the DDOS operating system.

Table 4.2 shows the two different enclosures that we measured. Each enclosure can support various capacity SATA drives. Based on vendor specifications, the drives we used have power usage of about 6–8W idle, 8–12W active, and less than 1W when spun-down. Controllers communicate with the enclosures via Serial Attached SCSI (SAS). Large system configurations can support more than fifty enclosures attached to a single controller, which can host more than a petabyte of physical capacity and tens of petabytes of logical, deduplicated capacity.

	ES20	ES30
Ship Year	2006	2011
# Disks	16	15
# SAS Controllers	2	2
# Power Supplies	2	2
# Fans	2	4

Table 4.2: Enclosure hardware summary

Experimental setup We measured controller power consumption using a Fluke 345 Power Quality Clamp Meter [39], an in-line meter that measures the power draw of a device. The meter provides readings with an error of $\pm 2.5\%$. We measured enclosure power consumption using a WattsUP Pro ES [136], another in-line meter, with an accuracy of $\pm 1.5\%$ for measured value plus a constant error of ± 0.3 watt-hours. All measurements were done within a data center environment with room temperature held between $70\,^{\circ}\mathrm{F}$ and $72\,^{\circ}\mathrm{F}$.

We connected the controllers and enclosures to the meters separately, to measure their power. Thus we present component's measurement separately, rather than as an entire system (e.g., a controller attached to several enclosures). The meters we used allowed us to measure only entire

device power consumption, not individual components (e.g., each CPU or HBA) or data-center factors (e.g., cooling or network infrastructure). We present all measurements in watts and all results are an average of several readings with standard deviations less than 5%.

Benchmarks For each controller and enclosure, we measured the power consumption when idle and when under several backup workloads. Each workload is a standard, reproducible workload used internally to test system performance and functionality. The workloads consist of two clients connecting over a 10 GigE network to a controller writing 36 backup streams. Each backup stream is periodic in nature, where a full backup image is copied to the controller, followed by several incremental backups, followed by another full backup, and so on. For each workload we ran 42 full backup generations. The workloads are designed to mimic those seen in the field for various backup protocols.

	WL-A	**WL-B**	**WL-C**
Protocol	NFS	OST	BOOST
Chunking	Server	Server	Client

Table 4.3: Backup workloads used

We used the three backup protocols shown in Table 4.3. Clients send backup streams over NFS in WL-A, and over Symantec's OST in WL-B. In both cases, all deduplication is performed on the server. WL-C uses, BOOST [28], an EMC backup client that performs stream chunking on the client side and sends only unique chunks to the server, reducing network and server load. To measure the power consumption of a fully utilized disk subsystem, we used an internal tool that saturates each disk.

4.3 Discussion

We present our analysis for a variety of configurations in three parts: isolated controller measurements, isolated enclosure measurements, and whole-system analysis using controller and enclosure measurements.

4.3.1 Controller Measurements

We measured storage controller power consumption under three different scenarios: idle, loaded, and power managed using processor-specific power-saving states.

Controller idle power A storage controller is considered idle when it is fully powered on, but is not handling a backup or restore workload. In our experiments, each controller was running a full, freshly installed, DDOS software stack, which included several small background daemon processes. However, as no user data was placed on the systems, background jobs such as garbage collection, were not run. Idle power consumption indicates the minimum amount of power a non-power-managed controller would consume when sitting in the data center.

It is commonly assumed that disks are the main contributor to power in a storage system. As shown in Table 4.4, the controllers can also consume a large amount of power. In the case of

	DD880	DD670	DD860	DD990
Idle Power (W)	555	225	261	778

Table 4.4: Idle power consumptions for storage controllers

DD990, the power consumption is almost equal to that of 100 2TB drives [58]. This is significant because even a controller with no usable disk storage can consume a lot of power. Yet, the performance of the controller is critical to maintain high deduplication ratios, and necessary to support petabytes of storage—requiring multiple fast CPUs and lots of RAM. These high idle power-consumption levels are well known [72]. Although computer component vendors have been reducing power consumption in newer systems, there is a long way to go to support true power proportionality in computing systems; therefore, current idle controller power levels must be factored into future designs.

■ **Observation 1:** *The idle controller power consumption is still significant.*

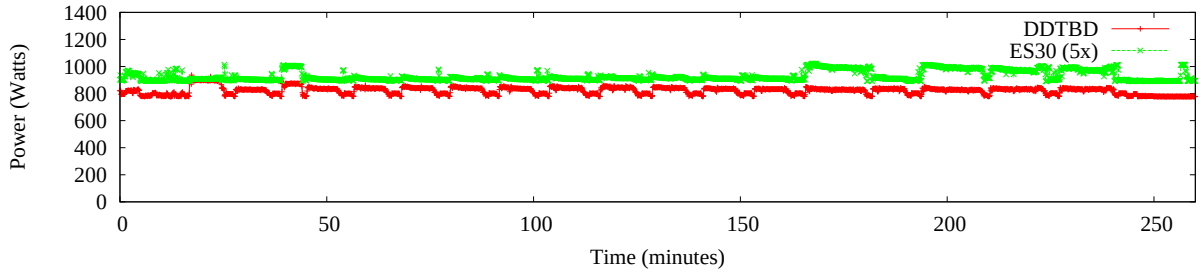
Table 4.4 shows a large difference in power consumption between controllers. DD990 consumes almost 3.5× more power than DD670. Here, difference is largely due to the different hardware profiles. DD990 is a more powerful, high-end controller with significantly more CPU and memory, whereas DD670 is a low-end model. However, this is not the case for the power differences between DD880 and DD860. DD880 consumes more than twice the power as DD860, yet Table 4.1 shows that their hardware profiles are fairly similar. The amount of CPU and memory plays a major role in power consumption; however, other factors such as the power efficiency of individual components also contribute. Unfortunately, our measurement methodology prevented us from identifying the internal components that contribute to this difference. However, part of this difference can be attributed to DD860 being a newer model with hardware components that consume less power than older models.

To better compare controller power consumption, we normalized the power consumption numbers in Table 4.4 to the maximum usable physical storage capacity. The maximum capacities for the DD880, DD670, DD860, and DD990 are 192TB, 76TB, 192TB, and 1152TB, respectively. This gives normalized power consumption values of 2.89W/TB for DD880, 2.96W/TB for DD670, 1.35W/TB for DD860, and 0.675W/TB for DD990. Although the normalized values are roughly the same for DD880 and DD670, the watts consumed per raw byte trends down with newer generation platforms.

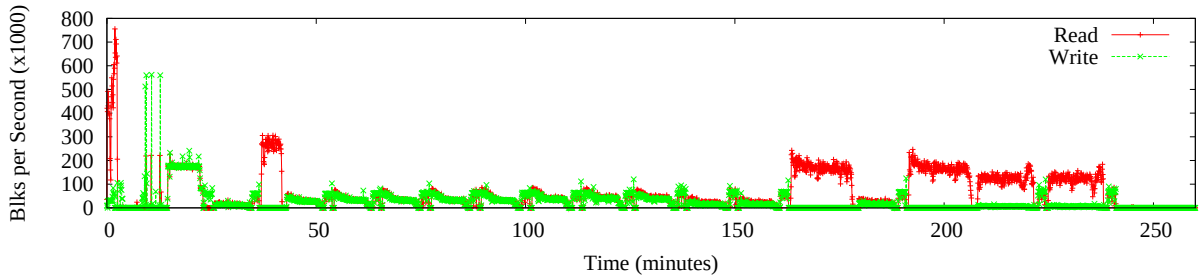
■ **Observation 2:** *Whereas idle controller power consumption varies between models, normalized watts per byte goes down with newer generations.*

Controller under load We measured the power consumption of each controller while running the aforementioned workloads. Each controller ran the DDFS deduplicating file system [152] and all required software services. Services such as replication were disabled. The power consumed under load approximates the power typically seen for controllers in-use in a data center. The workloads used are performance-qualification tests that are designed to mimic real customer workloads, but do not guarantee that the controllers are stressed maximally.

Figure 4.2(a) shows the power consumed by DD990 while running the WL-A workload. The maximum power consumed during the run was 937W, which is 20% higher than the idle power



(a) Power consumption



(b) I/O statistics

Figure 4.2: Power consumption and I/O statistics for WL-A on DD990. There are 5 ES30 enclosures attached to it.

consumption. Since the power only increased 20% when under load, it may be more beneficial to improve idle consumption before trying to improve active (under load) consumption.

	DD880	**DD670**	**DD860**	**DD990**
WL-A	44%	24%	58%	20%
WL-B	58%	29%	61%	36%
WL-C	56%	28%	57%	23%

Table 4.5: Power increase ratios from idle to loaded system

Table 4.5 shows the power increase percents from idle to loaded across controller and workload combinations. Several combinations have an increase of less than 30%, while others exceed 50%. Unfortunately, our methodology did not allow us to identify which internal components caused the increase. One noticeable trend is that the increase in power is mostly due to the controller model rather than the workload, as DD880 and DD860 always increased more than DD670 and DD990.

■ **Observation 3:** *The increase in controller power consumption under load varies much across models.*

I/O statistics from the disk sub-system help explain the increases in controller power consumption. Figure 4.2(b) shows the number of blocks per second read and written to the enclosures attached to DD990 during WL-A. We see that a higher rate of disk I/O activity generally corresponds to higher power consumption in both the controller and disk enclosures. Whereas I/Os require the controller to wait on the disk sub-system, they also increase memory copying activity, communication with the sub-system, and deduplication fingerprint hashing.

Power-managed controller Our backup systems perform in-line, chunk-based deduplication, requiring significant CPU and RAM amounts to compute and manage hashes. As the data path is highly CPU-intensive, applying DVFS techniques during backup—a common way to manage CPU power consumption—can degrade performance. Although it is difficult to throttle CPU during a backup, the backup processes are usually separated by large idle periods, which provide an opportunity to exploit DVFS an other power-saving techniques.

Intel has introduced a small set of CPU power-saving states, which represent a range of CPU states from fully active to mostly powered-off. For example, on the Corei7, C1 uses clock-gating to reduce processor activity, C3 powers down L2 caches, and C6 shuts off the core's power supply entirely [124]. To evaluate the efficacy of the Intel C states on an idle controller, we measured the power savings of the deepest C state. Unfortunately, DD990 was the only model that supported the Intel C states. We used a modified version of CPUIDLE to place DD990 into the C6 state [75]. In this state, DD990 saved just 60W, a mere 8% of total controller power consumption. This finding suggests that DVFS alone is insufficient for saving power in controllers with today's CPUs and a great deal of RAM. Moreover, deeper C states incur higher latency penalties and slow controller performance. We found that the latencies made the controller virtually unusable when in the deepest C state.

■ **Observation 4:** *Placing today's Intel CPUs into deep C state saves only a small amount of power and significantly harms controller performance.*

4.3.2 Enclosure Measurements

We now analyze the power consumption of two generations of disk enclosures. Similar to Section 4.3.1, we analyzed the power consumption of the enclosures when idle, under load, and using power-saving techniques.

Enclosure idle power An enclosure is idle when it is powered on and has no workload running. The idle power consumption of an enclosure represents the lowest amount of power a single enclosure and the housed disks consume without power-management support. Figure 4.3 shows that an idle ES20 consumes 278W. The number of active enclosures in a high-capacity system can exceed 50, so the total power consumption of the disk enclosures alone can exceed 13kW.

We found that the enclosures have very different power profiles. The idle ES20 consumes 278W, which is 55% higher than the idle ES30, at 179W. We believe that newer hardware largely accounts for this difference. For example, it is well known that power supplies are not 100% efficient. Modern power supplies often place guarantees on efficiency. One standard [1] provides an 80% efficiency guarantee, which means the efficiency will never go below 80% (e.g., for every 10W drawn from the wall, at least 8W is usable by components attached to the power supply). The ES30 has newly designed power supplies, temperature-based fan speeds, and a newer internal controller, which contribute to this difference.

■ **Observation 5:** *The idle power consumption varies greatly across enclosures with new ones being more power efficient.*

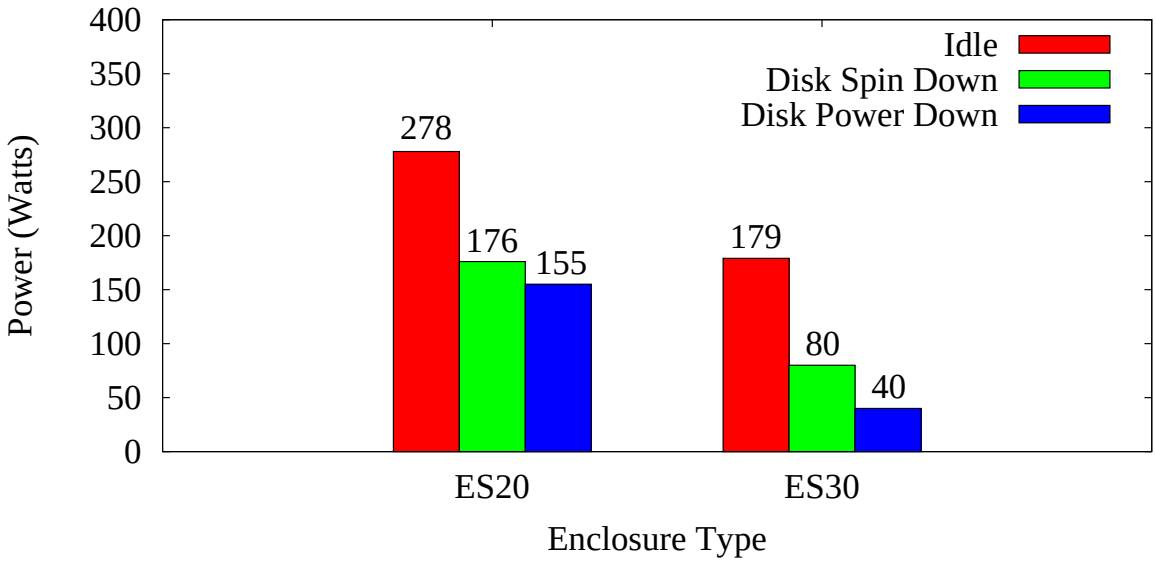


Figure 4.3: Disk power down vs. spin down. ES20 and ES30 are specified as in Table 4.2.

Enclosure under load We also measured the power consumption of each enclosure under the workloads discussed in Section 4.2. We considered an enclosure under load when it was actively handling an I/O workload.

As shown in Figure 4.2(a), the total power consumption of the five ES30 enclosures connected to DD990, processing WL-A, increased by 10% from 900W when idle to about 1kW. Not surprisingly, Figure 4.2(b) shows that an increase in enclosure power correlates with an increase in I/O traffic. Percentages for the other enclosure and workload combinations ranged from 6–22%.

Our deduplicating file system greatly reduces the amount of I/O traffic seen by the disk sub-system. As described in Section 4.2, we used an internal tool to measure the power consumption of a fully utilized disk sub-system. Table 4.6 shows that ES20 consumption grew by 22% from 278W when idle to 340W. ES30 increased 15% from 179W idle to 205W. Interestingly, these increases are much smaller than those observed for the controllers under load in Section 4.3.1.

■ **Observation 6:** *The consumption of the enclosures increases between 15% and 22% under heavy load.*

Power managed enclosure We compared the power consumption of ES20 and ES30 using two disk power-saving techniques: power-down and spin-down. With spin-down, the disk is powered on, but the head is parked and the motor is stopped. With power-down, the enclosure's disk slot is powered off, cutting off all drive power.

As shown in Figure 4.3, the relative power savings of the ES20 and ES30 are quite different. For ES30, spin-down reduced power consumption by 55% from 179W to 80W. For ES20, the power dropped by 37% from 278W to 176W. Although the absolute spin-down savings was roughly 100W for both enclosures, power-down was much more effective for ES30 than ES20. Power-down for ES30 reduced power consumption by 78%, but only 44% for ES20. As mentioned in Section 4.2, each disk consumes less than 1W when spun-down. However, for both ES20 and ES30, power-down saved more than 1W per disk compared to spin-down.

	ES20	ES30
Idle Power (W)	278	179
Max Power (W)	340	205

Table 4.6: Max power for enclosures ES20 and ES30

■ **Observation 7:** *Disk power-down may be more effective than disk spin-down for both ES20 and ES30.*

Looking closer at the ES20 power savings, the enclosure actually consumes more power than the disks it is housing (an improvement opportunity for enclosure manufactures). With all disks powered down, ES20 consumes 155W, which is more than the 123W saved by powering down the disks (consistent with disk vendor specs).

■ **Observation 8:** *Disk enclosures may consume more power than the drives they house. As a result, effective power management of the storage subsystem may require more than just disk-based power-management.*

We observed that an idle ES30 enclosure consumes 64% of an idle ES20, while a ES30 in power-down mode consumes only 25% of the power of an ES20 in power-down mode. This suggests that newer hardware's idle and especially power-managed modes are getting better.

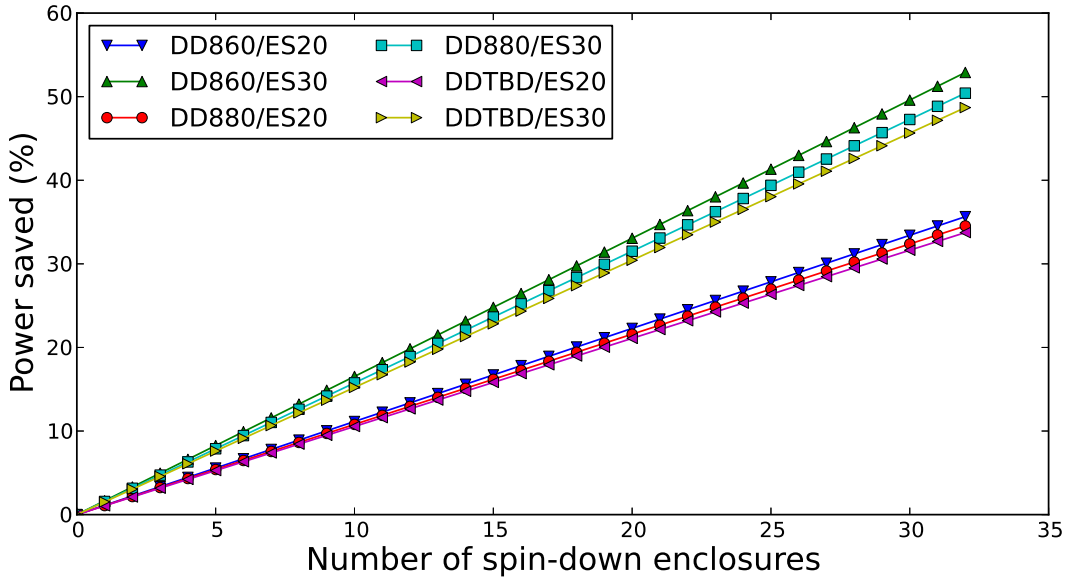
4.3.3 System-Level Measurements

A common metric for evaluating a power management technique is the percentage of total system power that is saved. We measured the amount of power savings for different controller and enclosure combinations using spin-down and power-down techniques. We considered system configurations with an idle controller and 32 idle enclosures (which totals 512 disks for ES20 and 480 disks for ES30) and we varied the number of enclosures that have all their disks power managed. We excluded DD670 because it supports only up to 4 external shelves.

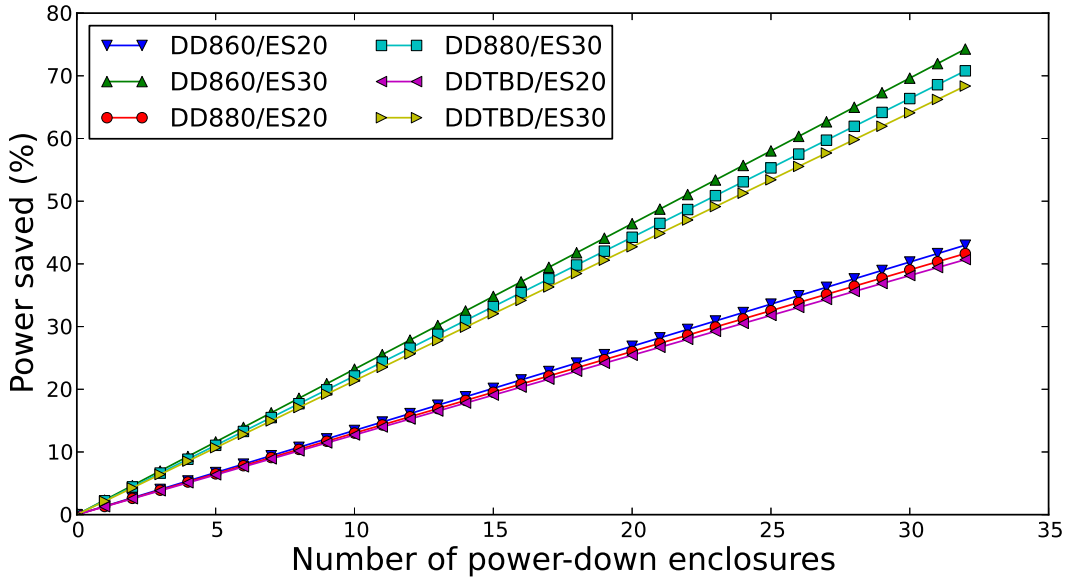
Figure 4.4 shows the percentage of total system power saved as the number of enclosures with power-managed disks was increased. In Figure 4.4(a) disks were spun down, while in Figure 4.4(b) disks were powered down. We found that it took a considerable number of power-managed disks to yield a significant system power savings. In the best case with DD860 and ES30, 13 of the 32 enclosures must have their disks spun down to achieve a 20% power savings. In other words, over 40% of the disks must be spun down to save 20% of the total power. In the worse case with DD990 and ES20, 19 of the 32 enclosures must have their disks spun down to achieve a 20% savings. This scenario required almost 60% of the disks to be spun down to save 20% of the power. Only two of our six configurations were able to achieve more than 50% savings even when all disks were spun down. These numbers were improved when power down is used, but a large number of disks was still needed to achieve significant savings.

■ **Observation 9:** *To save a significant amount of power, many drives must be in a low power mode.*

The limited power savings is due in part to the controllers consuming a large amount of power. As seen in Section 4.3.1, a single controller may consume as much power as 100 disks. Additionally, as shown in Section 4.3.2, disk enclosures can consume more power than all of the drives they



(a) Disk Spin Down vs. Power Savings Percentage



(b) Disk Power Down vs. Power Savings Percentage

Figure 4.4: Total system power savings using disk power management

house, and the number of enclosures must scale with the number of drives in the system. These observations indicate that for some systems, even aggressive disk power management may be insufficient to save enough power and that power must be saved elsewhere in the system (e.g., reducing controller and enclosure power consumption, new electronics, etc.).

4.4 Conclusions

We presented the first study of power consumption in real-world, large-scale, enterprise, disk-based backup storage systems. Although we investigated only a handful of systems, we already uncovered a three interesting observations that may impact the design of future power-efficient backup storage systems.

(1) We found that components other than disks consume a significant amount of power, even at large scales. We observed that both storage controllers and enclosures can consume large amounts of power. For example, DD990 consumes more power than 100 2TB drives and ES20 consumes more power than the drives it houses. As a result, future power-efficient designs should look beyond disks to target controllers and enclosures as well.

(2) We found a large difference between idle and active power consumption across models. For some models, active power consumption is only 20% higher than idle, while it is up to 60% higher for others. This observation indicates that existing systems are not achieving energy proportionality [3, 9, 48, 128, 132], which states that systems should consume power proportional to the amount of work performed. For some systems, we found a disproportionate amount of power used while idle. As backups often run on particular schedules, these systems may spend a lot of time idle, opening up opportunities to further reduce power consumption.

(3) We discovered large power consumption differences between similar hardware. Despite having similar hardware specifications, we observed that the older DD880 model consumed twice as much idle power as the newer DD860 model. We also saw that an idle ES20 consumed 55% more power than an idle ES30. This suggests that the power profile of an existing system can be improved by retiring old hardware with newer, more efficient hardware. We hope to see continuing improvements from manufacturers of electronics and computer parts.

We have discussed the power consumption in enterprise-scale backup storage systems. Next, we explore in details the performance and energy consumption of systems software in Chapter 5.

Chapter 5

On the Energy Consumption and Performance of Systems Software

The carbon footprint of the IT industry, though 2% of the world economy, is estimated to be equal to that of the entire aviation industry [22]. Energy consumption is emerging as a critical issue in the design of computing systems [15, 33, 51, 60, 64, 91, 154]. The goals of energy-aware system design include saving energy without sacrificing performance, and supporting flexible, dynamic trade-offs between energy consumption and performance. Accurate models of energy consumption and performance provide a foundation for the design of energy-aware systems.

A large portion of the energy consumed by IT infrastructure is due to desktop machines and commercial servers [24]. Moreover, the total amount of electronic data stored world-wide is rising exponentially. Thus, it is desirable to develop highly scalable solutions that are significantly better than today's solutions.

In prior work, we analyzed the energy and performance profiles of server workloads, such as Web servers, email servers, database servers, and file compression [72, 119]. We discovered large deviations for both performance and energy consumption—as much as 10 times—suggesting that there are significant opportunities to save energy and improve performance. Our past work considered those systems only as black-boxes and reported their performance and energy consumption without a deeper understanding of the exact reasons for those deviations.

Seeking a better understanding of the system internals of these workloads, we tried to identify their internal behavior, so we could build advanced controllers to better manage both energy and performance. Unfortunately, our initial attempts to identify these systems using traditional linear-systems identification techniques resulted in poor models with low prediction accuracy (under 50%).

In this chapter, we shed considerable light on the complexities underlying systems-software energy consumption and performance. In particular, we present an in-depth experimental evaluation of the energy consumption and performance of a relatively simple yet familiar file-compression workload as a representative workload involving both substantial CPU usage and disk I/O. We also analyze the effects of several input parameters, including choice of compression algorithm, compression level, file type, persistent storage media (e.g., SATA, SAS, and SSD), CPU Dynamic Voltage and Frequency Scaling (DVFS) level, and disk I/O scheduler—all under the Linux operating system.

Our experimental results show that energy consumption and performance are unexpectedly

complex and cannot be easily modeled using standard system-identification techniques. We identify several factors that contribute to this complexity, in terms of nonlinearity, instability, and multi-dimensionality. Our results suggest that hybrid discrete-continuous models [2, 56] may provide a suitable foundation for modeling and control of energy consumption and performance in energy-aware systems software.

The rest of the chapter is organized as follows. Section 5.1 considers related work. Section 5.2 provides the requisite background. Section 5.3 provides the motivation for this work. Section 5.4 presents our experimental setup and benchmarks. Section 5.5 contains our experimental results. We conclude in Section 5.6.

5.1 Related Work

This section places our work in the context of past work.

5.1.1 Energy Efficiency

Many energy-saving techniques have been developed at both the hardware and software levels. For example, virtualization allows multiple Operating Systems (OSs) to run on one server, sharing most of the resources, thereby reducing energy consumption. Moreover, there are energy-aware cache replacement algorithms [150], energy-aware task and interrupt management techniques [121], online learning-based power management [31], predictive data grouping and replication [33], and energy-aware file systems configuration pruning techniques [119]. Some modeling based approaches have been proposed by Isci, Sarikaya, and others [62, 111]. Some of our own past studies show significant energy savings possible in commodity Linux servers running common workloads such as Web, email, database, compression, etc. [72, 119]. Generally, optimal use of energy-saving techniques requires accurate models of system energy consumption with respect to appropriate parameters; the work described in Chapter 5 is a step towards the development of such models.

5.1.2 Energy Consumption of Data Compression

Our prior work, conducted by Kothiyal et al., evaluated energy consumption and performance of data compression on servers [72] and demonstrated that compression reduces energy consumption in some situations but not all. A careful application of compression can save energy in some cases by a factor of 10×, but a careless application of compression can easily waste energy and slow performance by 200×. In contrast to the work described in Chapter 5, our past study did not focus on accurate modeling of energy consumption and hence did not discuss system identification or analyze the behavioral characteristics of energy consumption and performance that make accurate modeling difficult.

5.2 Background

In this section, we describe background work in terms of compression algorithms (Section 5.2.1), I/O schedulers (Section 5.2.2), and power and energy consumption (Section 5.2.3).

5.2.1 Compression Algorithms

In Linux, there are three main compression utilities: `gzip`, `bzip2`, and `lzop`, each of which has compression levels ranging from level 1 to level 9. A higher level tries to achieve a better compression ratio at the expense of additional CPU cycles.

Gzip [40] is based on the DEFLATE algorithm, which is a combination of LZ77 and Huffman coding. Bzip2 uses the Burrows-Wheeler transform to convert frequently recurring character sequences into strings of identical letters and then applies a move-to-front transform and Huffman coding [18]. Lzop [95] uses the LZO algorithm instead and produces files a bit larger than Gzip's but with a lower CPU use. For `lzop`, compression levels 1 to 6 are identical.

5.2.2 I/O Schedulers

I/O scheduling has been studied aggressively [6, 8, 59, 69, 149] especially since the speed of disk lags far behind the speed of CPU and RAM.

Normally, a disk scheduler tries to maintain a balance between fairness, performance, and latency (or real time guarantees). Fairness guarantees that every process has fair share of the access to disk on a multi-user system. Performance requires the scheduler to serve requests predictably to save both time and energy. Latency means that any request must be served within a given time limit. There are four main I/O schedulers in Linux systems: (1) CFQ (the default), which emphasizes fairness; (2) ANTICIPATORY, which emphasizes performance; (3) DEADLINE, which is designed for low latency and real time access; and (4) NOOP, which is a simple first-come-first-served scheduler.

5.2.3 Power and Energy Consumption

In this subsection, we introduce the power and energy consumption patterns for both CPU and disk, since our workload is both CPU-intensive and disk-intensive.

The power consumed in a processor consists of three portions: dynamic power $P_{dynamic}$, static power P_{static}, and short-circuit power [88]. For Complementary Metal Oxide Semiconductor (CMOS) chips, dynamic power refers to the energy consumption in switching transistors, while static power refers to the flowing leakage current when a transistor is off. Short-circuit power is consumed only during signal transitions and is insignificant. The dynamic power is calculated as follows:

$$P_{dynamic} = C \times V^2 \times f \tag{5.1}$$

where C is the capacitance per cycle, V is the supply voltage and f is the processor clock frequency.

Although dynamic power is the primary source of power dissipation in CMOS chips [88], static power is becoming an important issue. Static power is computed as follows:

$$P_{static} = V \times I_{th} + V_{bs} \times (I_{jn} + I_{bn}) \tag{5.2}$$

where I_{th} is the sub-threshold leakage current, V_{bs} is the body bias voltage, and I_{jn} and I_{bs} are the drain and source to body junction leakage current, respectively.

Processors with Dynamic Voltage and Frequency Scaling (DVFS) are capable of operating at multiple frequency and voltage levels. Dynamic power is considered to be the dominant portion of the processor's energy consumption. As seen from Equation 5.1, $P_{dynamic}$ depends linearly on frequency and quadratically on voltage. However, operating at a lower voltage and frequency does not necessarily result in overall energy savings, as we see later in Section 5.5.3. The main reason is that when running at a lower frequency, it usually takes longer to accomplish the same work, which can increase the total energy consumption.

The power consumed by a Hard Disk Drive (HDD) follows the following equation:

$$P_{disk} = P_{spin} + P_{head} \tag{5.3}$$

where P_{spin} refers to the energy consumed by the spinning platter, and P_{head} refers to the energy consumption incurred by the movement of the disk head.

5.3 Motivation

Section 5.3.1 gives some background on system identification. Section 5.3.2 describes the problems we encountered when we tried to apply system identification techniques to model the energy consumption of our workload.

5.3.1 System Identification

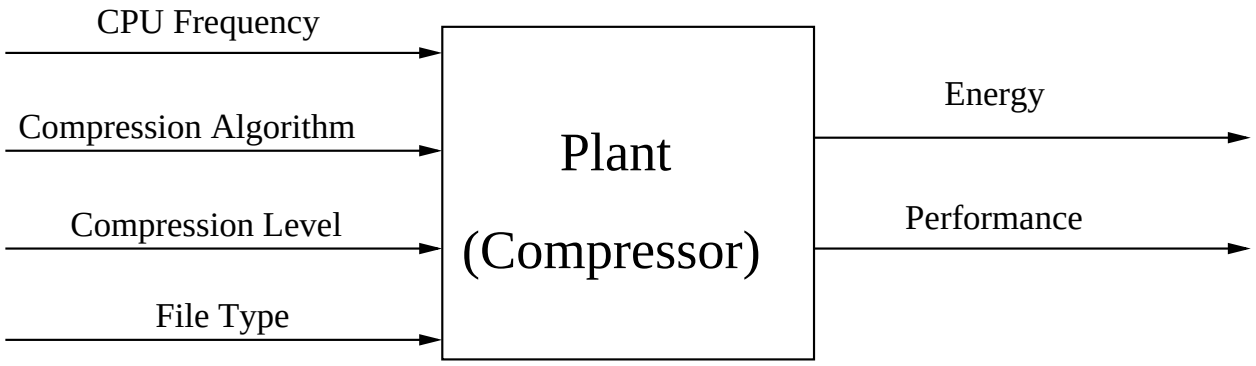


Figure 5.1: Plant: Compressor

System identification is the first step of control engineering that uses statistical methods to build models from observed behavior.

As shown in Figure 5.1, our system has four inputs: compression algorithm, compression level, file type, and CPU frequency. Our system has two outputs: energy and performance. Applying off-the-shelf technology for system identification, such as MATLAB's system identification toolbox [80] has considerable appeal, since one needs to know only the inputs and outputs. It does not require a detailed understanding of the system's behavior. By applying statistical techniques to data

collected from the target system, system identification attempts to construct a mathematical model of the relationships between inputs and outputs.

A typical workflow for system identification follows these four steps: (1) Specify the model in the form of inputs and outputs, and design experiments to collect data; (2) Apply the system identification algorithm to estimate the values of the coefficients of the model; (3) Verify the accuracy of the resulting model by evaluating it against additional measured data; (4) Decide whether the model is acceptable. If the prediction accuracy is unacceptably low, one or more steps in the workflow need to be revisited.

In our experiments, we used a traditional linear state-space model of the following form:

$$x(n+1) = Ax(n) + Bu(n) + Kw(n) \tag{5.4a}$$

$$y(n) = Cx(n) + Du(n) + w(n) \tag{5.4b}$$

where $u(n)$ are the inputs, $y(n)$ are the outputs, $x(n)$ the internal states of the plant, and $w(n)$ is a white Gaussian noise representing uncontrollable inputs and output measurement errors (e.g., errors introduced by the default system daemons) at time n. The parameter $x(n+1)$ denotes the next internal states of the plant. Matrices A, B, C, D, and K denote the significance or weight that each element in the input, output, and Gaussian noise have in determining the next state and output of the system.

5.3.2 Problems Encountered

Our system is a simple file compressor. System inputs x can be file type (ZERO, TEXT, BINARY, or RANDOM), compression level (1 to 9), compression algorithm (GZIP, BZIP2, LZOP, or NONE for no compression), and CPU frequency/voltage (eight available choices). We considered energy consumption and performance as the outputs y.

The system inputs and outputs must be quantified in order to apply system identification. Energy is measured in Watt-hours. Performance is measured as the number of files compressed per second. The CPU frequency is measured in Hertz. However, it is difficult to choose appropriate numerical values to represent file types, compression levels, and compression algorithms.

The compression level is numerical, but the level number is actually just a label (in other words, a name); the numerical value has no direct significance other than ordering. Similarly, file types and compression algorithms are naturally identified by discrete, non-numerical labels but must be represented numerically to apply the system identification algorithm. The numbers chosen are significant, because they must be related to the next states and outputs by Equation 5.4 for system identification to succeed and should not impose arbitrary quantitative relationships. However, we have no a-priori way of deciding what values to use.

We tried a simple linear approach using consecutive integers (e.g., 0 for NONE, 1 for GZIP, 2 for BZIP2 and 3 for LZOP), as well as other numbers and ordering. We also tried a non-linear approach, assigning each compression algorithm a number corresponding to its compression ratio; but the compression ratio varies with file type and hence is not a fixed value associated solely with the compression algorithm.

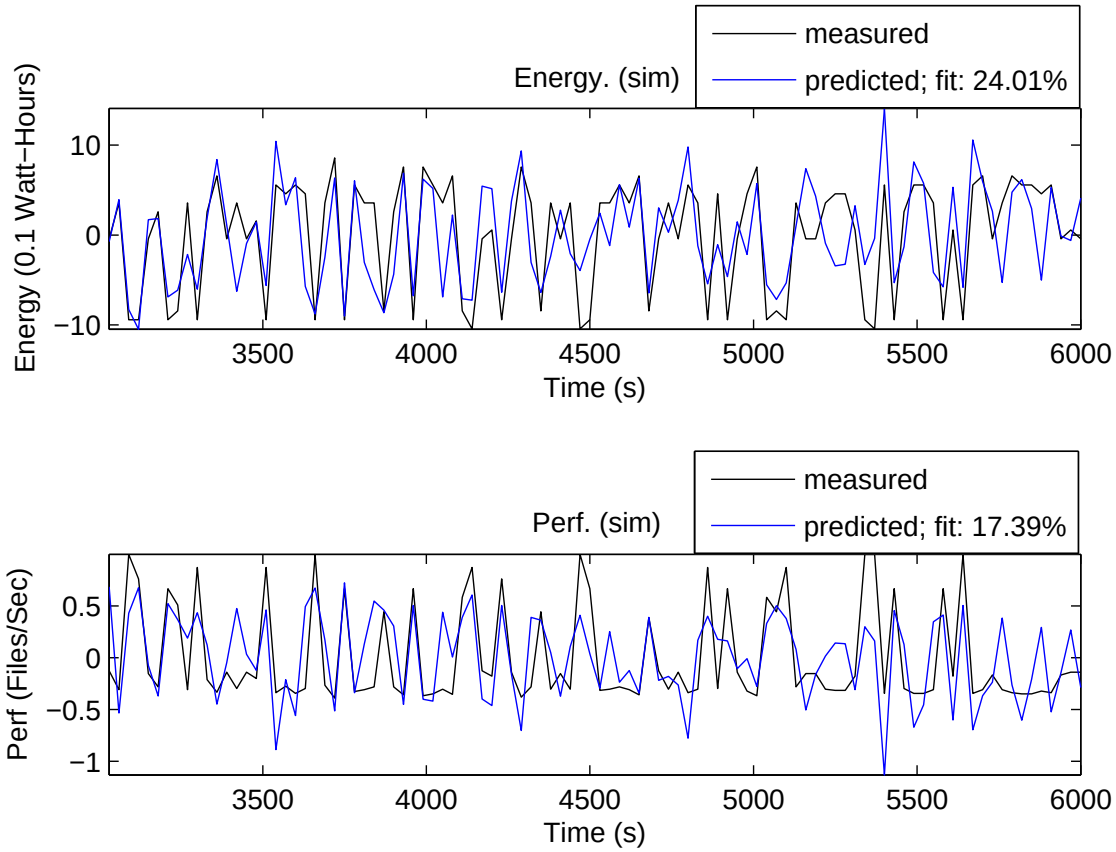


Figure 5.2: A typical example for poor accuracy. The two inputs are File Type and CPU Frequency. Compression algorithm is fixed to be gzip along with its default compression level. The two outputs are energy and performance which are normalized to be zero-mean.

In conclusion, labels are similar to the discrete states of a finite automaton. In our case, they represent different modes of system behavior; that is, they represent the modes of a hybrid automaton. Any attempt to give them a numerical meaning is doomed to fail.

We prepared two data sets of the same size to identify the system. One data set is used to estimate the parameters of the model using least-squares techniques; the other is used to evaluate the quality of the model fit. Accuracy is the percentage of model fit. We applied the MATLAB's system identification tool-box to learn Single-Input-Single-Output (SISO) and other system models. However, we achieved only limited accuracy, less than 50% in overall. A typical error graph appears in Figure 5.2.

This was clearly insufficient as a basis to design a controller. In order to better understand the causes of the problem, and to find ways of splitting the nonlinear behavior into segments that can be more accurately modeled as linear systems, we decided to study the system's energy consumption and performance in more detail.

5.4 Methodology

This section details our experimental setup and benchmarks.

5.4.1 Experimental Setup

We conducted our experiments on a Dell PowerEdge R710 server consisting of one quad-core Intel® Xeon™ Nehalem CPU with a maximum frequency of 2.395GHz with dynamic frequency and voltage scaling (DVFS) support: 7 different frequencies at a difference of 133MHz each without the Turbo Mode, and 8 different frequencies at a difference of 1MHZ for the top 2 frequencies (as Linux reports) and a difference of 133MHZ for the remaining 7 frequencies with the Turbo Mode on. The machine has 24GB RAM, out of which we used only 2GB to force I/O to take place. The server has two 146GB Seagate SAS disks with 15,000RPM rotation speed and a 16MB cache, two 250GB internal Fujitsu SATA disks with 7,200 RPM rotation speed and 16MB cache, and one 80GB Intel SSD disk model SSDSA2MH080G1C5. We ran all of our benchmarks on all of these three different kinds of disk drives. The server was running the Linux 2.6.18 kernel with the `acpi_cpufreq` module installed to enable software control of the CPU frequency.

We connected the server to a WattsUP Pro ES in-line power meter [136], which measures the energy drawn by a device plugged into the meter's receptacle. The power meter uses non-volatile memory to store measurements every second. Its resolution is 0.1 Watt-hours (1 Watt-hour = 3,600 Joules). The accuracy is $\pm 1.5\%$ of the measured value plus a constant error of ± 0.3 Watt-hours. Its resolution for power measurements is 0.1 Watts. We used the `wattsup` Linux utility to download the recorded data from the meter over a USB interface to the test machine.

We conducted 216 combinations of experiments (repeated five times each), and collected a large data set: 4,810,320 data points in total for a single run. Running one complete set of benchmarks took about 15 calendar days to complete.

To automate the measurements, we developed a tool called auto-ebench, written in Perl and Bash, that helped us benchmark the energy and power consumption under different scenarios while launching `vmstat` to record the number of block reads and block writes. We measured the total number of block reads and writes at the whole-system level; this saved us significant time and effort.

5.4.2 Benchmarks

The workload for each test is to compress 20 identical files with 20 threads concurrently, and write the compressed files to disk. Each file is 65MB. Several factors influence energy consumption for data compression, as we will discuss in Section 5.5.3. In order to fully explore these factors and their interactions, we conducted experiments for each combination. Specifically, we consider the following factors: persistent storage media (SAS disk, SATA disk, and SSD disk), I/O scheduler (anticipatory, CFQ, deadline and NOOP), compression algorithm (gzip, bzip2, and lzop) and compression level (1–9), and file type (text, binary, and random). We ran the above workload for each combination of these factors. Between each compression level, we inserted some sleeping intervals, so that each experiment for each compression level started at the same exact time. The elapsed time for compression plus the sleeping interval was the same and fixed during each compression level, in order to align the graphs for each compression level. Auto-ebench is responsible for repeatedly

launching the experiments and recording the results multiple times and under multiple scenarios. Our experiments follow this pattern unless otherwise noted.

We ran all the tests five times and computed the 95% confidence intervals using the Student-t distribution. The error bars shown in our graphs are the half widths of the 95% confidence intervals. We used version 1.3.5 of `gzip`, version 1.0.3 of `bzip2`, and version v1.02rc1 of `lzop`.

The I/O scheduler can be set per device and is easy to configure. In order to set the I/O scheduler, we write the desired scheduler name to `/sys/block/$dev/queue/scheduler` and launch the experiments after that.

We ran the tests on the specified disk drive, formatted with Ext3 file system and mounted using the default options. To avoid caching effects, we unmounted the file system after each test iteration to flush the data in memory to disk. Our measurements include this flushing time.

5.5 Evaluation

In this section, we provide evaluation and deep analysis for the energy consumption pattern of our file-compression workload. Sections 5.5.1, 5.5.2, and 5.5.3 focus on non-linearity, instability, and multi-dimensionality, respectively.

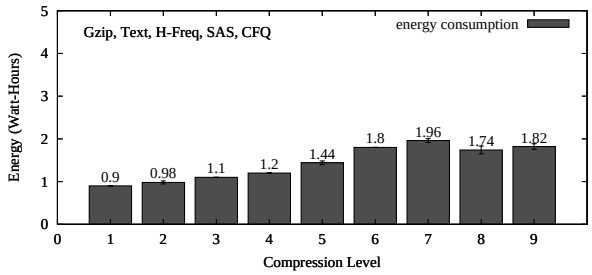
5.5.1 Nonlinearity

For compression algorithms, a higher compression level usually means a better compression ratio (CR). Table 5.1 shows the CR for all algorithms and levels.

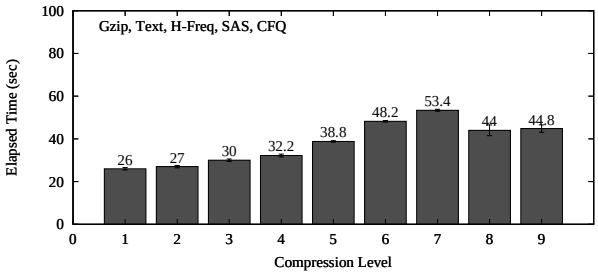
Although it is true that a higher compression level generally commits fewer blocks to disk for the same workload and hence might save energy due to reduced I/O activity, the overall energy consumption might not follow the same pattern. One possible reason is that the CPU may have to perform a lot more work in order to achieve a better CR, which takes longer time and consumes more energy. The actual energy consumed under certain workloads is in fact a trade-off between these factors. Therefore, as we can see from Figure 5.3, which presents measurements for `gzip`, `bzip2`, and `lzop`, the energy consumption is not a linear function of the compression level. Moreover, it is also not monotonically increasing with the compression level. For example, in Figure 5.3(b), energy consumption peaks at level 7, then unexpectedly drops at levels 8 and 9.

Comparing the graphs in the left and right columns of Figure 5.3, we also observe that the energy consumption for the whole system depends heavily on the total elapsed time during the compression period [27, 105].

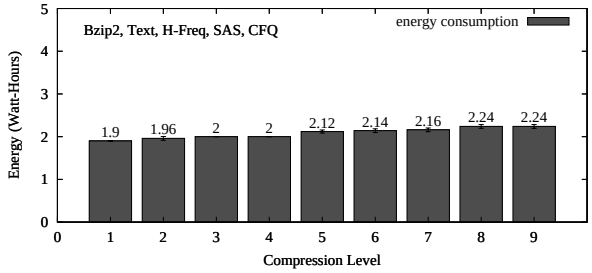
As we can see from Figure 5.3(a), in the case of `gzip`, the energy consumption goes up non-linearly and then goes down slightly as the compression level increases. Figure 5.3(b) shows that the elapsed time follows the same trend. In the case of `bzip2` as shown in Figure 5.3(c), the energy consumption is relatively stable, increasing only slightly across all 9 compression levels, which suggests that a balance between the CPU energy consumption and disk energy consumption has been achieved. The elapsed time, shown in Figure 5.3(d), follows the same pattern. With `lzop`, as shown in Figure 5.3(e), the energy consumption is the same for the first six identical compression levels and then increases monotonically but non-linearly. This reflects that for the last three compression levels, due to the longer elapsed time, the entire system (including the disk drive, even when it is just spinning, not reading or writing) is consuming more power at higher



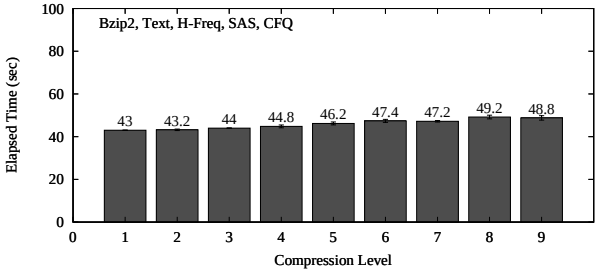
(a) **Energy** consumption in each compression level of gzip



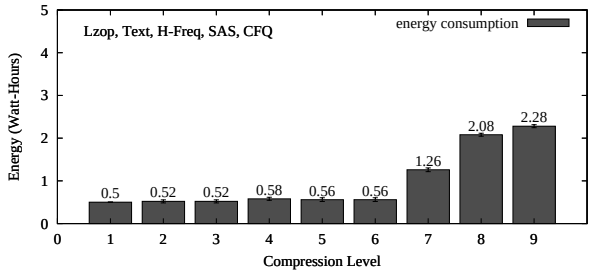
(b) **Time** taken in each compression level of gzip



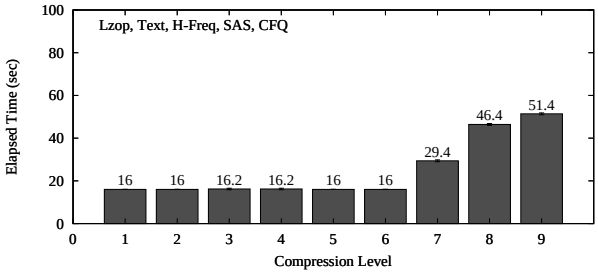
(c) **Energy** consumption in each compression level of bzip2



(d) **Time** taken in each compression level of bzip2



(e) **Energy** consumption in each compression level of lzop



(f) **Time** taken in each compression level of lzop

Figure 5.3: An example combined graph for illustrating nonlinearity. Experiments compressing text files using the highest CPU frequency, SAS disks, and the CFQ I/O scheduler. In 5.3(a), the x axis denotes the compression level and the y axis denotes Watt-hours (equals to 3,600 Joules). In 5.3(b), the unit for Elapsed Time is seconds. This representation is kept the same for 5.3(c), 5.3(d), 5.3(e), and 5.3(f).

compression levels even though slightly fewer blocks are written to disk. Figure 5.3(f) show that the elapsed time strictly follows the same pattern.

In summary, it is clear that the energy consumption and elapsed time relate non-linearly and in some cases non-monotonically with the compression level. Consequently, controlling the system's energy usage by adjusting the compression level is complex.

Tool	File Type		
	Text	Binary	Rand
gz-1	3.61	2.14	1.00
gz-2	3.77	2.18	1.00
gz-3	3.90	2.21	1.00
gz-4	4.18	2.26	1.00
gz-5	4.35	2.30	1.00
gz-6	4.43	2.32	1.00
gz-7	4.45	2.33	1.00
gz-8	4.46	2.33	1.00
gz-9	4.46	2.33	1.00
bz-1	4.72	2.38	0.99
bz-2	5.02	2.45	0.99
bz-3	5.18	2.53	0.99
bz-4	5.28	2.57	0.99
bz-5	5.36	2.60	0.99
bz-6	5.40	2.64	0.99
bz-7	5.44	2.65	1.00
bz-8	5.49	2.67	1.00
bz-9	5.50	2.69	1.00
lzo-(1∼6)	2.82	1.77	1.00
lzo-7	3.80	2.15	1.00
lzo-8	3.84	2.16	1.00
lzo-9	3.84	2.17	1.00

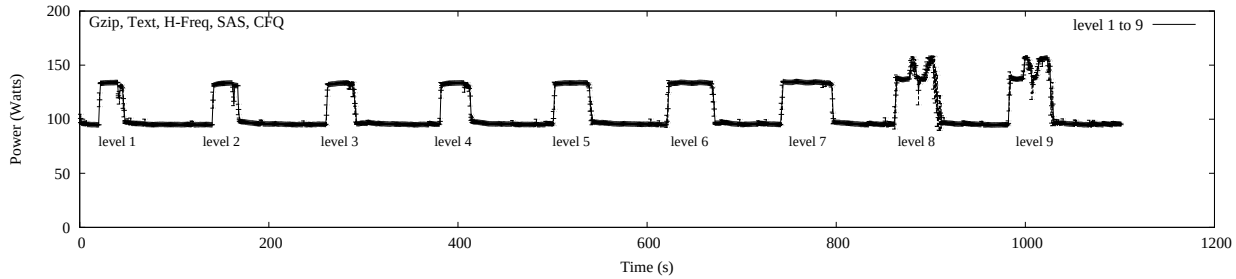
Table 5.1: Compression ratios achieved by various compression utilities and levels

5.5.2 Instability

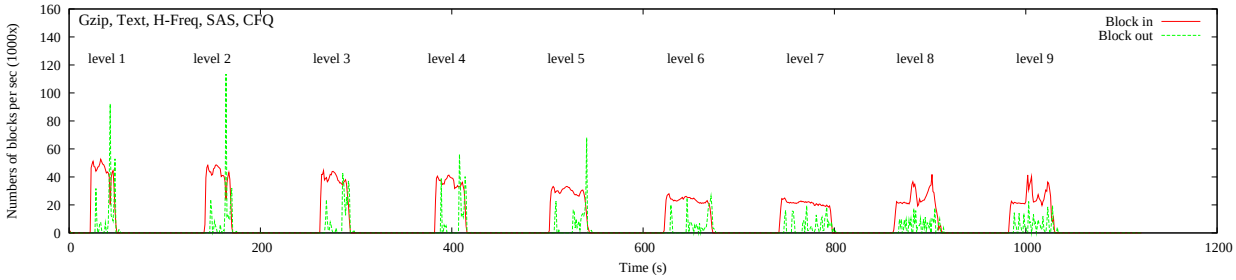
This section examines how the power consumption varies during each run. We found that in some cases, the power consumption response is unstable and fluctuates significantly, as we can see from Figures 5.4, 5.6, and 5.7. This should be taken into consideration when designing power-aware systems.

Our experiments revealed that the cause of those fluctuations lies in the interleavings between disk reads and writes when the CPU frequency is maintained at the same level. We discuss this in more detail below.

For `gzip` (Figure 5.4(a)), the power consumption response is relatively stable from level 1 to level 7. However, it becomes unstable in levels 8 and 9. Furthermore, Figure 5.4(b) reveals that the rate at which blocks are read exhibits the same pattern of stability in levels 1 to 7 and fluctuation at levels 8 and 9. Looking at Figure 5.4(b) in detail, especially in levels 8 and level 9, it also reveals more frequent interleavings between block reads and writes. For levels 1 through 4, there are just small fluctuations in power consumption towards the end of the compression job. A similar pattern appears in the interleavings between block reads and writes for levels 1 to 4. Moreover, the stable response in levels 5, 6, and 7 suggests equally distributed interleavings between the rates of block reads and writes. We believe that when block reads and writes are interleaved beyond a certain level, I/O scheduler algorithms (and possibly algorithms inside the disk) begin to break down and



(a) Power consumption response for each level of compression of gzip



(b) Rate of block reads and writes for each level of compression of gzip

Figure 5.4: Relationship between the rates of block reads/writes and power consumption of gzip. The y axis is in units of thousands of reads/writes. The CPU frequency is set to the highest frequency in the above experiments. One can see that there are fluctuations in levels 8 and 9.

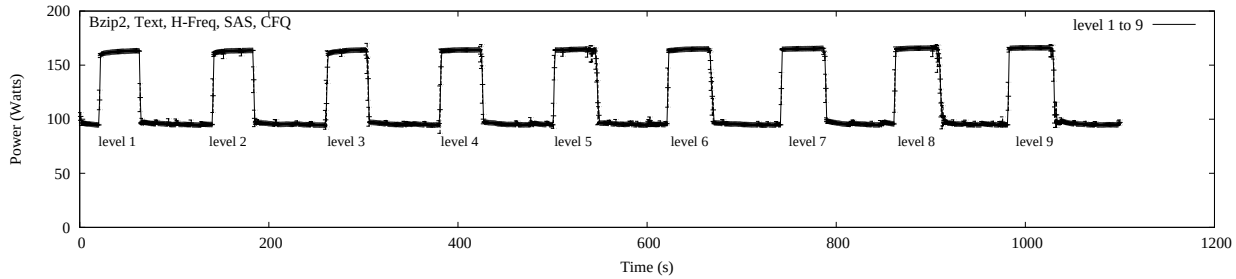
their efficiency goes down considerably as a result.

For `bzip2` (Figure 5.5(a)), the power consumption response is relatively stable. In Figure 5.5(b), we can see clearly that the rate of disk block reads is maintained at a stable level, and the rate at which disk blocks are written is equally distributed throughout the compression period. This leads the power consumption response to be stable.

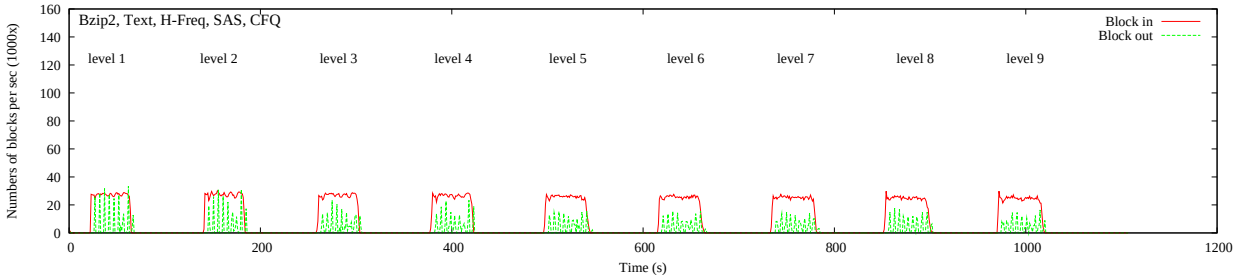
For `lzop` (Figure 5.6(a)), the power consumption response follows a different pattern compared with the previous two scenarios. We can see from Figure 5.6(b) that for the first six levels, the I/O rate is much higher than in the remaining levels. However, the run is shorter in terms of elapsed time, as we can see from the width of the active intervals, and the interleavings between the rates of block reads and writes are in some degree not equally distributed across the compression level, resulting in a few fluctuations towards the end of each compression level. For levels 8 and 9, since the interleavings are equally distributed, the power response is relatively stable. The fluctuations in level 7 suggest there exists unequally distributed interleavings between the rates of disk reads and writes.

An even more complicated example appears in Figure 5.7(a). In this scenario, random files are being compressed and SATA is the persistent storage media. The power response is extremely unstable in each compression level. The interleavings between the rates of block reads and writes are ill regulated, as we can see from Figure 5.7(b). This suggests that the harder the file is to compress (e.g., high entropy), the less predictable the performance and energy consumption are. There is no simple way to model systems exhibiting such complex and diverse behavior.

We conclude that the power response exhibits instability in many cases. This contributes to the



(a) Power consumption response for each level of compression of bzip2



(b) Rate of block reads and writes for each level of compression of bzip2

Figure 5.5: Relationship between the rates of block reads/writes and power consumption of bzip2. The CPU frequency is set to the highest frequency in the above experiments. One can see that the power response is stable for each compression level.

complexity of the energy usage of the system and makes controlling a serious challenge.

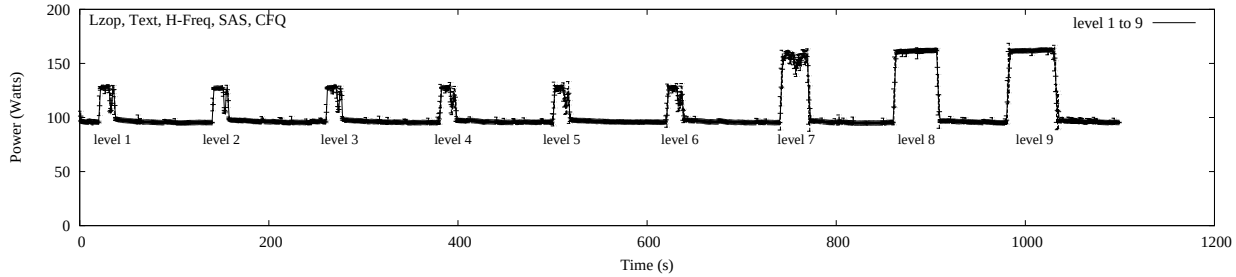
5.5.3 Multi-Dimensionality

In this section, we illustrate the dependence of the energy consumption for our system on several factors, such as CPU frequency, compression algorithm and level, file type, persistent storage media, and disk I/O scheduler.

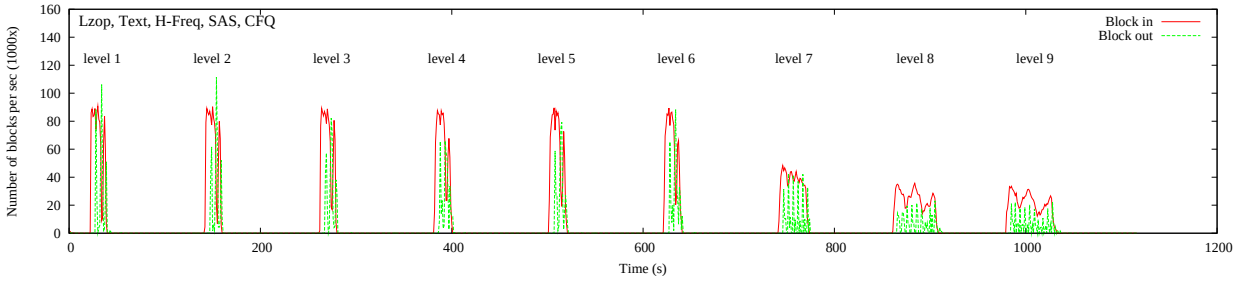
The compression algorithm is clearly an important factor of energy consumption here, as we have already seen in Figure 5.3. For example, `bzip2` takes much longer time to compress than `lzop` does. Thus, `bzip2` usually takes more energy to compress than `lzop` does.

One might expect that a lower CPU frequency will result in lower energy consumption. However, as we can see from Figures 5.8(a) and 5.8(b), that is not necessarily true. With lower CPU frequency, the energy consumption is actually increased for all the compression levels. The reason is that when the CPU frequency is lower, it takes longer to finish the compression, which generally results in a higher total energy consumption. We can also see from Figure 5.8 that for both the highest frequency and the lowest frequency, the consumed energy increases as a function of compression level. However, there is also a possibility that when the CPU frequency is lower, the rate at which the CPU compresses data in the blocks will be closer to the rate at which the disk drive produces blocks. If this happens, it can save energy at the end, since there is no wasted energy.

The disk I/O scheduler influences the order of disk writes and hence may affect the energy consumption. Figure 5.9 shows the energy consumption with 4 different I/O schedulers. We can

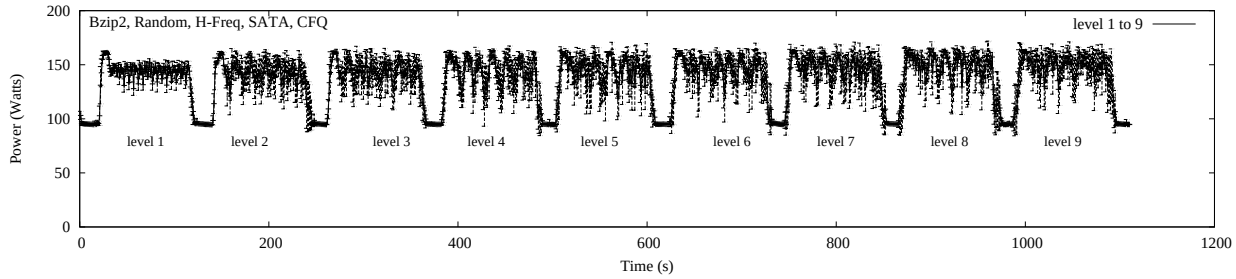


(a) Power consumption response for each level of compression of lzop

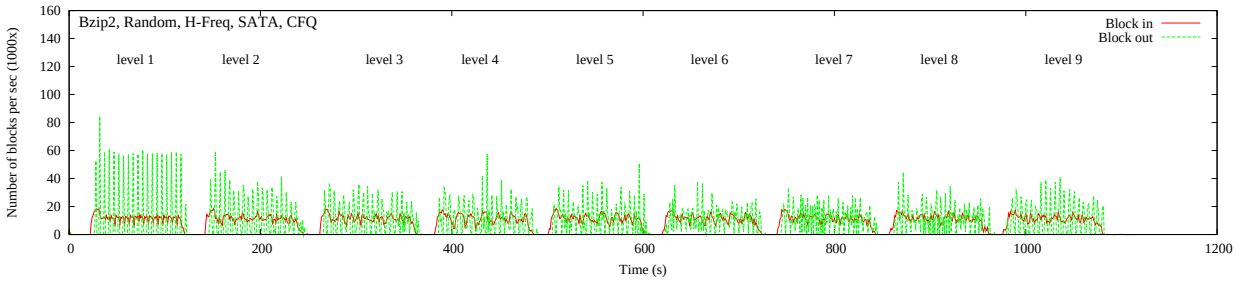


(b) Rate of block reads and writes for each level of compression of lzop

Figure 5.6: Relationship between the rates of block reads/writes and power consumption of lzop. The CPU frequency is set to the highest frequency in the above experiments. One can see fluctuations from levels 1 to 7.

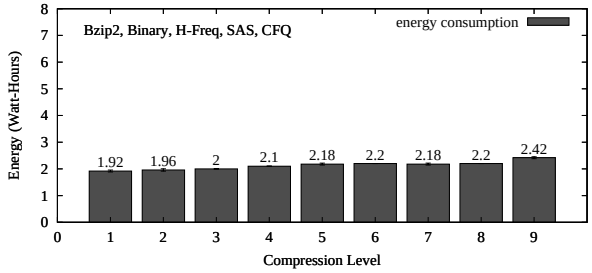


(a) Power consumption response for each level of compression of bzip2 with SATA and random files

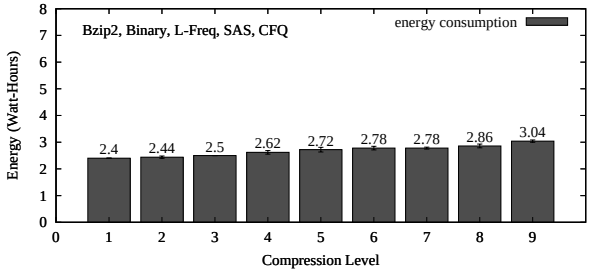


(b) Rate of block reads and writes for each level of compression of bzip2 with SATA and random files

Figure 5.7: An even more complex example. The CPU frequency is set to the highest frequency in the above experiments. One can see large fluctuations during every compression level.

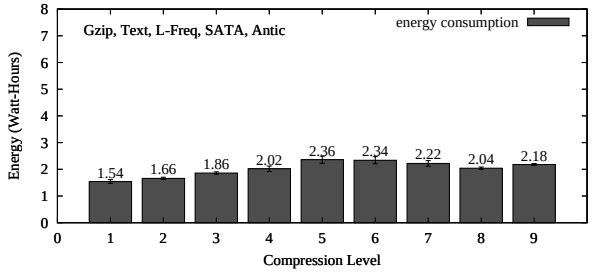


(a) **Energy** consumption in highest frequency with each compression level

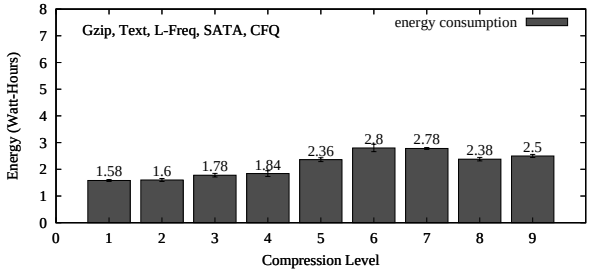


(b) **Energy** consumption in lowest frequency with each compression level

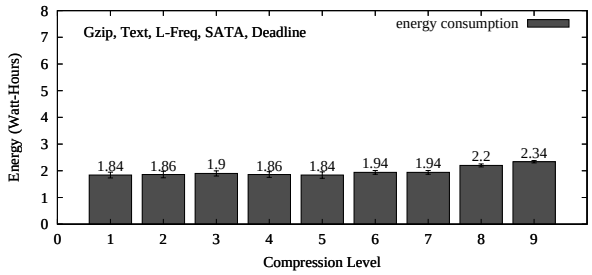
Figure 5.8: Energy consumption at the highest and lowest CPU frequencies



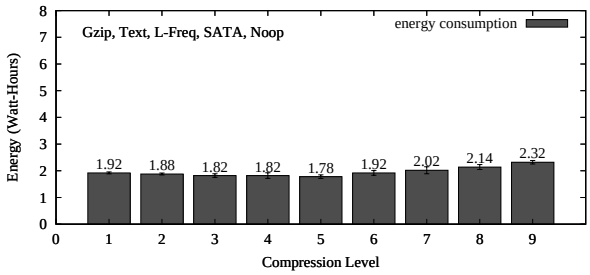
(a) **Energy** consumption with anticipatory for each compression level



(b) **Energy** consumption with CFQ for each compression level



(c) **Energy** consumption with deadline for each compression level

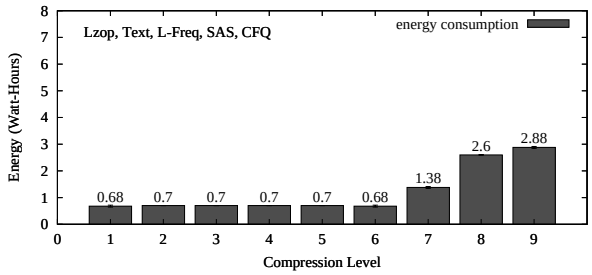


(d) **Energy** consumption with NOOP for each compression level

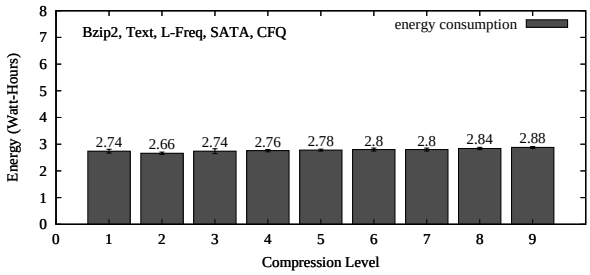
Figure 5.9: Energy consumption under 4 different I/O schedulers

see that anticipatory and CFQ have largely the same effect, while deadline and NOOP also have similar effect to each other but different from anticipatory or CFQ. As the unit for the y axis is Watt-hours, the difference in energy consumption between anticipatory and CFQ is actually significant, especially for larger workloads.

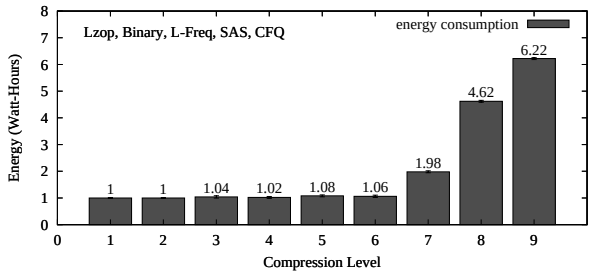
The file type affects different compression strategies for each compression algorithm and hence plays a role in energy consumption. The left column of Figure 5.10 shows the energy consumption of the compression workload for different file types. We see that the workload with binary files consumes more energy than the workload with text files when other parameters are the same; this makes sense because text files have more common patterns that can be compressed (e.g., lower entropy). Also for text and binary files, more energy is consumed with compression level 9 than with



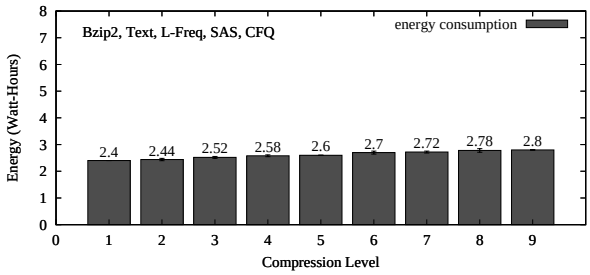
(a) **Energy** consumption of text files with each compression level



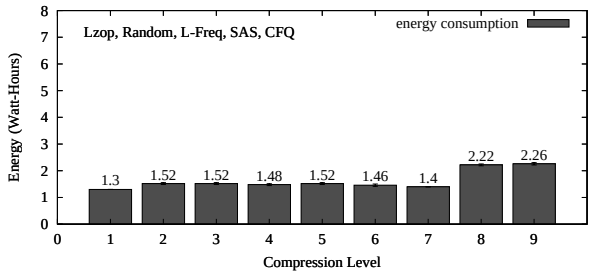
(b) **Energy** consumption of SATA



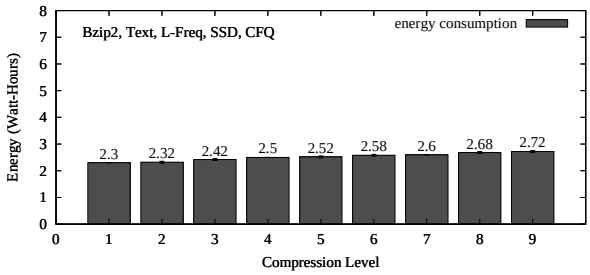
(c) **Energy** consumption of binary files with each compression level



(d) **Energy** consumption of SAS



(e) **Energy** consumption of random files with each compression level



(f) **Energy** consumption of SSD

Figure 5.10: Energy consumption for different File types and disk types

other compression levels. Surprisingly, for random files, level 8 turns out to be the most energy-consuming one, instead of level 9. We conclude that the file type affects the energy consumption response in a manner that is not easy to predict, and an approach involving adaptive feedback control may thus be required.

Different disk types usually have different electronics and firmware, different physical features, and different storage strategies. This should affect energy consumption. The right column of Figure 5.10 shows the energy consumption of the compression workload for different persistent storage media. As expected, SAS is generally faster than SATA, so the workload runs faster and consumes less energy, 2–12% less. SSD is the fastest storage media among the three, consuming the least energy, 3–5% less than SAS and 6–16% less than SATA. This is because an SSD contains no energy-consuming moving parts (cf. Equation 5.3) and stores data on non-volatile flash memory chips using a Flash Translation Layer (FTL) that allows the linear device to look like a traditional disk. These results also show that the workload is not completely CPU bound, even though it is

CPU intensive.

In summary, we observe that the total energy consumption of computer systems follows a complicated pattern, because the energy consumption for each subsystem contributes to it. This suggests that instead of trying to develop system-level energy models purely in a bottom-up fashion, a more practical approach may be to use machine learning methods in the development of such models to guide the design of energy-aware systems.

5.6 Conclusions

Accurate models of energy consumption and performance are vital for the design and implementation of energy-efficient systems. Our detailed experimental results show that the behavior of these quantities is far more complicated than one might expect, even for a relatively simple workload such as data compression. The complexity is reflected in nonlinearity, instability, and multi-dimensionality. These factors must be considered in the design of energy-efficient systems.

Although we have measured and analyzed the effects of several factors, there may be other important factors to consider, depending on the system, such as the workload itself, and even the server and machine-room temperatures.

We have discussed the energy and performance details of systems software. Next, we explore the GreenDM's details in Chapter 6.

Chapter 6

GreenDM: A Versatile Tiering Hybrid Drive for the Trade-Off Evaluation of Performance, Energy, and Endurance

In previous Chapters, we analyzed system performance, energy, and power under various conditions, and we described how the device endurance interfacts with the other dimensions. In this Chapter, we discuss the details of GreenDM.

The rest of the chapter is organized as follows. Section 6.1 illustrates the design and implementation details. Section 6.2 presents the evaluation results and discussions. Section 6.3 shows the related work. Section 6.4 discusses the limitations the future direction of our work. Section 6.5 concludes the chapter.

6.1 Design and Implementation

We describe GreenDM's design and implementation in this section. Section 6.1.1 presents the design goals. Section 6.1.2 shows the system architecture. Section 6.1.3 details the design. Section 6.1.4 describes our power-management techniques. Section 6.1.5 describes the endurance model used for the trade-off study. Section 6.1.6 presents the implementation details.

6.1.1 Design Goals

The work was motivated by several concerns in storage systems. With the advent of SSDs, there were more opportunities to tackle these concerns. Specifically, with GreenDM, our design goals were as follows:

1. **Hybrid Drive:** we want to build a tiering hybrid drive with efficient data management and additional power management, where the SSD was used as primary storage, as the benchmark system.

2. **Trade-offs Study:** we would like to come up with a per-device endurance metric to help study the trade-offs.

3. **Versatility:** we want to have versatile policies so that the system can adapt to different workloads.

To meet our desired design goals, GreenDM (1) migrates hot data to the primary device (SSD), and migrates cold data to the secondary device (HDD)—useful in workloads that exhibit hot/cold I/O patterns; (2) decouples the logical storage address space from the physical one to allow flexible data placement; (3) decouples the migrations between the SSD and the HDD to improve concurrency between CPU and I/O; (4) optimizes the data management by serving I/O requests directly from RAM instead of the SSD whenever possible; (5) throttles migrations between the SSD and the HDD to control the overhead and effectiveness of migrations; (6) uses the lower-power SSD over the HDD and spins down the HDD when it is idle for a sufficient amount of time; and (7) is implemented in the Linux DM framework to be scalable. Note that we do not aim for super fast performance, or super efficient energy consumption, or optimized device endurance. Instead, the techniques we used above just serve the purpose of building a hybrid drive for us to study quantitatively the trade-offs among performance, energy, and endurance.

To help study the trade-offs among performance, energy, and endurance, GreenDM counts and utilizes the number of SSD reads and writes and the number of HDD start-stop (spin-up/down) cycles to estimate the devices' endurance. SSDs especially can wear out quickly and become less durable [120], and a mechanical disk drive can only be spun down and up for a limited number of cycles [64].

To achieve the versatility goal, GreenDM supports several controllable parameters so that the system can be tuned to different workloads.

6.1.2 Architecture

We implemented GreenDM in the Linux DM framework, to benefit from its scalability and flexibility. Figure 6.1 presents our system's architecture. We detail GreenDM's internals in the following sections. "Linear" is another existing DM target that linearly maps from the virtual storage address space to the physical one. GreenDM is scalable: it can be easily configured to use multiple drives with minor code change. However, to better study and understand the fundamental behavior of our tiering hybrid drive, we used a two-drive setup in this paper: one SSD as the primary drive and one HDD as the secondary drive.

6.1.3 Data Management

GreenDM tries to keep hot data in the SSD so that the system benefits most from the SSD's superior performance and efficient energy consumption. To achieve this, GreenDM migrates hotter data to the SSD, and migrates colder data to the HDD as the working set changes over time. To guarantee the correctness of moving data around, GreenDM uses a mapping table to keep track of data movement. Figure 6.2 illustrates GreenDM's data management. GreenDM divides the Virtual Block Address (VBA) space and the Logical Block Address (LBA) space into *extents* that are multiple of the (4KB) page size for efficient data management. The Extent Size (ES) is a configurable parameter, but once configured, the size is fixed for the lifetime of the DM instance. Our extents are atomic units of data migration. GreenDM maintains the mapping information, from the VBA

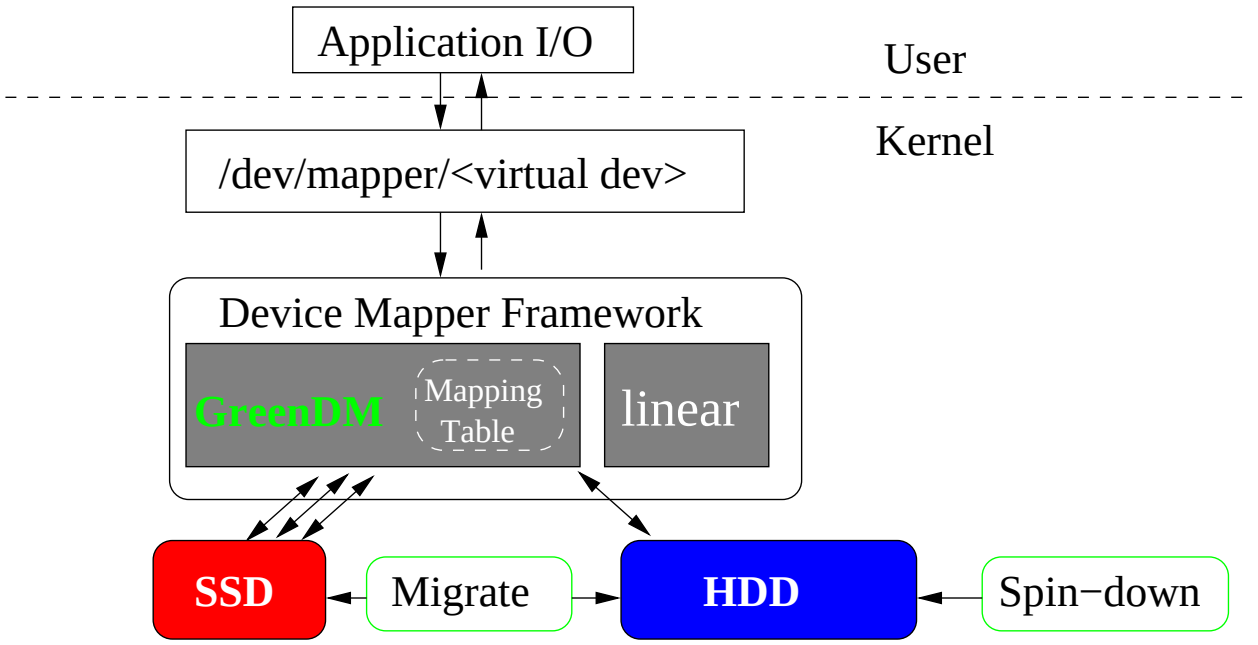


Figure 6.1: **GreenDM Architecture**. *The shaded rectangles are DM targets, usually implemented as loadable kernel modules.*

to the LBA, in the mapping table; each table entry maps from one Virtual Extent (VE) to one Logical Extent (LE). Data migration involves migrating LEs between the SSD and the HDD, and then updating the mapping table accordingly.

Mapping table The mapping table is a core data structure in GreenDM, as shown in Figure 6.2. It has four fields: LE ID, State, Usage Counter, and Time-stamp of the latest access. The LE ID identifies one LE. State represents the accessing state of each extent. The usage counter represents the number of total accesses. The time-stamp records the latest access of one specific extent. GreenDM populates the mapping table lazily. With a new virtual drive, the table starts empty. GreenDM creates the mappings in accordance with the workload. Compared to fully initializing the table with linear mapping, this approach provides more flexibility to data migration, especially when the workload is light. GreenDM uses a bit in the State field of each table entry to indicate if the entry is empty or not. GreenDM uses a bitmap to indicate whether the LEs on both drives are occupied or not. The mapping table and the bitmap together comprise the metadata of GreenDM. Whenever a new VE is accessed, GreenDM first allocates one free LE and then sets the corresponding mapping entry and the bitmap field properly. To locate a free LE, GreenDM always starts from the lower LBAs so that it improves the SSD's utilization. To accelerate this operation, GreenDM maintains an in-memory only free list for free LEs on the SSD.

Data separation GreenDM separates hot I/Os from cold I/Os based on their access frequencies, and stores them separately to best utilize the tiering hybrid device. Temporal locality suggests that once an extent is accessed, it is likely that the extent will be accessed again soon. In our case, active I/Os are first served through the primary drive (i.e., SSD) and the mapping is established accordingly. Inactive I/Os are kept on the secondary drive (i.e., HDD). GreenDM is designed this

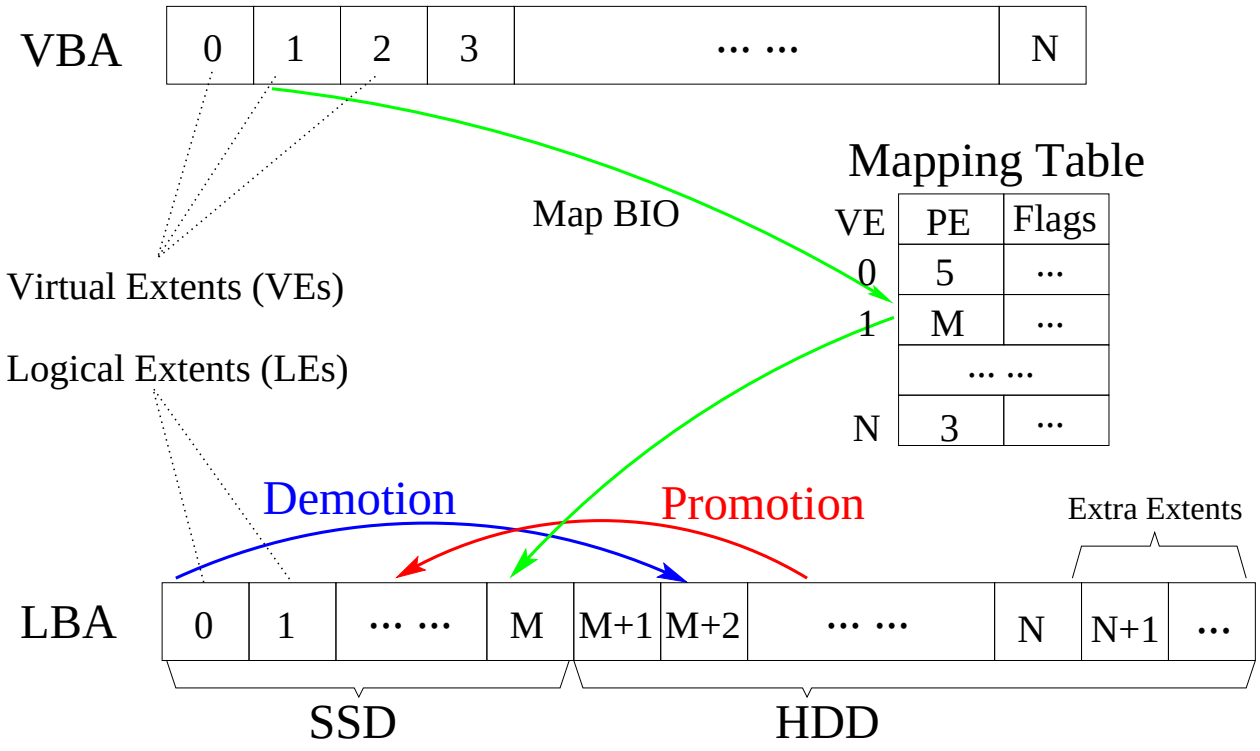


Figure 6.2: **Data Management in GreenDM**. *The green arrows show the address translation work-flow from one Virtual Extent (VE) to one Logical Extent (LE).*

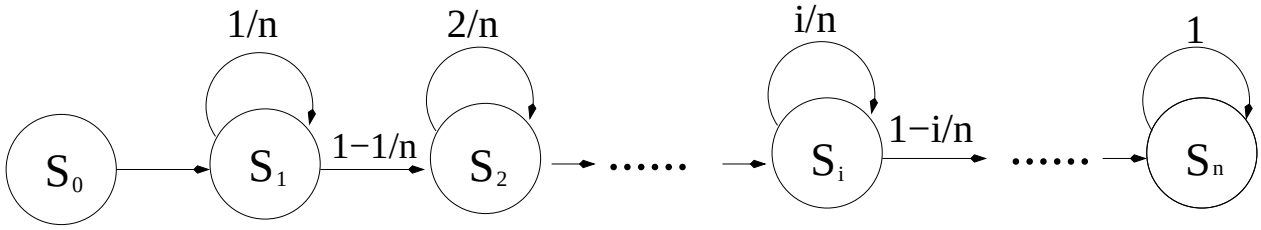


Figure 6.3: System State Transitions

way so that hot I/Os can mainly be served by the fast but smaller SSD, and cold I/Os are held on the slow but larger HDD drive.

Decoupled mapping GreenDM decouples the VBA space from the LBA space by mapping the VBA access to start from the Lowest Numbered LBA (LNL) of the SSD and the HDD drives first. If the mapping table is initially empty, then the decoupled mapping mechanism would shift the block access to the LNL in a monotonically decreasing manner for random read workload. Note that the mapping itself does not assume anything, we use random workload just to illustrate its mapping effect. We developed the theory to explain this behavior.

Suppose the virtual device has n virtual blocks (VBs) in total. As shown in Figure 6.3, we use S_i to represent the system state when it has i ($0 \leq i \leq n$) mapped logical blocks (LBs). The system can only go from state S_i to state S_{i+1} ($1 \leq i+1 \leq n$) or stay at the same state. When it transits,

one VB will be mapped to the $(i+1)^{th}$ LB, which will then be accessed only once. Since the workload is random, when there is one VB access, it has the same probability to be any one of the n VBs. Therefore, the probability for the system state to transit from S_i to S_{i+1} $(1 \leq i+1 \leq n)$ is $1 - \frac{i}{n}$, and the probability for the system state to stay the same is $\frac{i}{n}$. This is in fact a Markov Chain as shown in Figure 6.3.

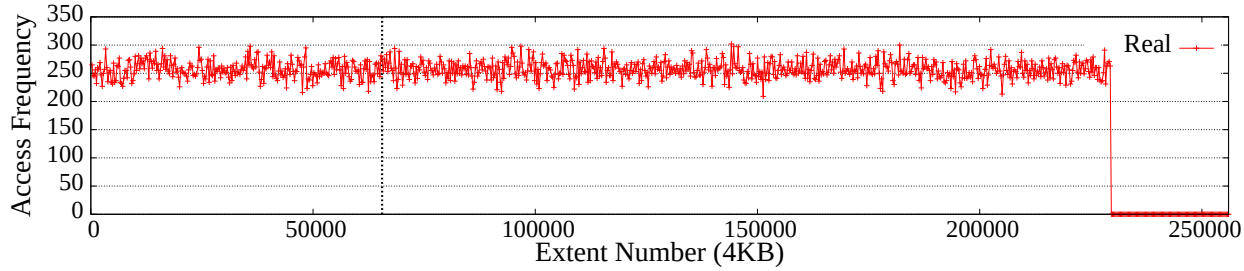
We use T_i $(1 \leq i \leq n)$ to represent how long the system stays in state S_i. Here, $\overline{thpt}_i$ represents the average throughput when the system stays in state S_i. We use B_s to represent the block I/O size. For simplicity, we let the I/O size be equal to the extent size. We use $Eacc(B_i)$ to represent the expected access frequency of block i. In state S_i, each physical block, starting from 1 to i, has the same probability of being accessed. We can then have the following equations:

$$Eacc(B_i) = 1 + \sum_{j=i}^{n} \frac{1}{j} \times M_j (1 \leq i \leq n) \tag{6.1}$$

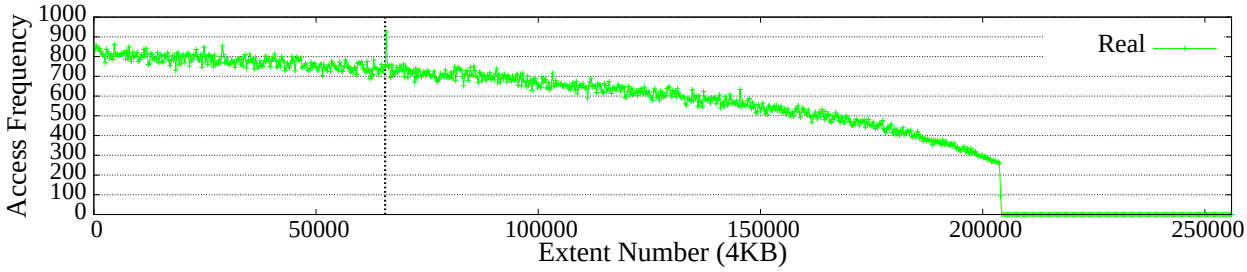
$$M_i = \frac{T_i \times \overline{thpt}_i}{B_s} (1 \leq i \leq n) \tag{6.2}$$

$$Eacc(B_i) > Eacc(B_{i+1}) (1 \leq i+1 \leq n) \tag{6.3}$$

We can see from Equation 6.3 that the expected access frequency of block i is larger than that of block $i+1$. It shows that GreenDM should shift the block access to the LNL in a monotonically decreasing manner.



(a) **Linear**



(b) **GreenDM**

Figure 6.4: LBA Space Access Frequency for **Linear** and **GreenDM**. *Figures are plotted to scale: 256 extents are merged into one point. The vertical line represents the boundary between the SSD capacity and the HDD capacity. Since the workload does not cover every single VB, some of VBs are not accessed. GreenDM observes this because starting from the LNL, every single LB is being mapped and accessed, while Linear does not.*

The DM Linear is a one-to-one direct mapping. For random read workloads, the uniformly distributed access from the VBA space is directly mapped to the LBA space. Thus, the access pattern on the LBA space should follow a uniform distribution. To verify the correctness of our analysis, we ran random read experiments with Filebench [35]. We set up a 1GB tiering hybrid drive with 256MB on the SSD and 768MB on the HDD for both GreenDM and Linear. We ran experiments for 20 minutes each and collected a block trace using `btrace`. The processed trace, seen in Figure 6.4, confirm our theory.

Data promotion To speed future accesses, promotions move hot LEs from the HDD to the SSD as the workload changes over time. To detect hot LEs on the HDD, GreenDM counts the number of I/O misses for every LE. An I/O is considered missed when the mapped LE resides on the HDD. A LE is considered hot if the number of I/O misses exceeds the Promotion Threshold (PT). GreenDM increases the LE miss count if two adjacent I/O accesses to the LE is within a Time Window (TW). Otherwise, the miss count starts over from beginning. The PT and the TW are both configurable parameters. Once GreenDM decides to promote a hot LE, it allocates a free LE on the SSD and enqueues the job to a promotion queue. GreenDM invokes a worker thread to keep dequeuing promotion jobs and copy data from source LEs to destination LEs synchronously. When the mapped LE of one VE is being promoted, accesses to the VE are suspended before being served; then GreenDM updates the mapping table. GreenDM cancels promotion attempts under any of the following conditions: (1) the SSD is full, because promotion requires free space in the SSD; (2) the metadata is being flushed to disk, because promotion has to update the metadata; (3) the Maximum Concurrent Migration Limit (MCML) is reached, because we throttle migration; or (4) there is concurrent access on the extent that is to be promoted, because the extent is already being accessed. Thus, instead of accessing the SSD, the HDD is accessed. This may delay HDD's spin-down and help maintain SSD's endurance, but increase access latencies.

Data demotion Demotion moves cold LEs from the SSD to free LEs on the HDD. There are different ways to perform data demotion. One approach is to evict SSD LEs instantly when promotions are taking place but there are no free extents on the SSD. This approach adapts well to the workload changes. However, it can prolong the promotion I/O latencies, which is undesirable. Another alternative is to schedule demotion as a periodic background job. However, this strategy is detrimental to energy efficiency because it has to wake up the HDD periodically. Instead, GreenDM schedules demotion in the background when the number of free LEs on the SSD drops to a configurable Low Threshold (LT). Once demotion is launched, it keeps demoting extents until the number of free LEs on the SSD reaches the configurable High Threshold (HT). The default value of HT is higher than the LT so that cold LEs are demoted efficiently in batch without constantly disturbing the HDD. When all LEs are mapped, GreenDM uses a small number of extra reserved extents in the HDD, as shown in Figure 6.2, to allow the demotion to find free LEs. Otherwise, data migration stalls if no free LE is found. The demotion thread uses the WSClock algorithm to find cold extents and updates the mapping table accordingly. GreenDM uses a device-mapper kernel thread called `dm_kcopyd`, which copies data between disk drives asynchronously.

Migration throttling GreenDM throttles data migrations to improve throughput. The mapping table has one field to count the number of accesses for each Physical Extent (LE). When the Promo-

Abbrev.	Name	Ex. Values
ES	Extent Size (in 4K units)	4K, 16K, 64K
PT	Promotion Threshold	1, 2, 4, 8
MCML	Maximum Concurrent Migration Limit	2, 4, 8, 16, 64
SP	Spin Down Policy	On, Off
LT	Low Threshold of demotion	32, 64
HT	High Threshold of demotion	64, 128
TW	Time Window length (sec)	30, 60

Table 6.1: **GreenDM Parameters and Abbreviations**

tion Threshold (PT) of one LE is reached, data promotion is attempted. The PT is configurable: (1) a larger PT can decrease the number of promotions and reduce the overhead, especially when there are lots of accesses; and (2) when the benefit of one promotion exceeds the overhead, a larger PT reduces the potential benefits. Migration is also throttled by the Maximum Concurrent Migration Limit (MCML). The MCML specifies the maximum concurrent promotions and demotions. The MCML is tunable: (1) a larger MCML value can promote hot I/Os to the SSD earlier and prepare free SSD extent slots earlier to benefit future accesses; and (2) a larger MCML value can potentially choke the system as ongoing migrations can freeze other I/O requests. Demotion tries to maintain [LT, HT] free extents in the SSD so that promotion can just use the free extent instead of waiting for demotion to proceed. Demotion is designed to decouple from promotion to improve interleaving between CPU and I/O.

Serving directly from RAM To save I/Os, GreenDM serves buffered I/O requests directly from RAM instead of the SSD in case of a successful promotion. The size of the RAM buffer is equal to the size of the hot LE. When a hot LE is being promoted, I/Os mapped to it will be pending before being served. A naïve approach to serve the pending I/Os is to first migrate one LE from the HDD to the SSD, and then access the SSD again to serve the pending I/O requests one by one. However, this approach triggers more SSD accesses than needed. Instead, GreenDM first reads the LE data from the HDD to RAM; then, GreenDM serves pending I/Os directly from RAM, before the LE data is written to the SSD. GreenDM invokes the DM API `bio_endio` to indicate that the I/O request was served. For each pending I/O, this saves one SSD I/O cycle by serving directly from RAM. This approach can save many SSD I/Os because when a LE is hot, it is likely to be accessed many times even during the short period of promotion. If there are more I/Os accessing the same LE while data is being flushed from RAM to SSD, GreenDM suspends these I/Os in the queue and serves them from the SSD as usual.

Versatility To enable adaptation to different workloads, GreenDM supports several configurable system parameters: ES, PT, MCML, SP (spin-down policy), LT, HT, and TW. Table 6.1 summarizes the parameters in more detail. All parameters can be set at the user level. The ES can be set in the GreenDM configuration file before the tiering hybrid drive starts service. The users can set the remaining parameters at any time by accessing the corresponding Linux `debugfs` entries.

6.1.4 Power Management

In addition to the above data-management techniques, our GreenDM manages the power consumption of the system to save energy. First, GreenDM saves energy simply by using the SSD in preference to the HDD. To further save power, when the secondary disk is idle, GreenDM spins down the drive [153]. The side effect of this spin-down is two-fold: (1) it takes time for a spun-down disk to spin back up, and (2) it reduces the HDD endurance if the HDD is spun up and down too frequently as each (mechanical) HDD has a limited number of start-stop cycles. GreenDM spins down the disk when it is idle for at least five seconds, configured by `hdparm`. We chose five seconds because it is the time it takes to spin down the HDD we used. The smaller the time-out latency is, the more aggressive the HDD spin-down policy is. When there is access on the spun-down disk, it spins up automatically.

6.1.5 Endurance Model

	Limits
SSD	36,500 GB writes
HDD	300,000 spin up/down cycles

Table 6.2: Devices Wear-out Limits.

GreenDM explores the endurance model for both the SSD and the HDD. For the HDD, GreenDM utilizes the number of start-stop cycles as the major factor towards endurance.

For the Flash-based SSD, it suffers from the endurance problem because Flash device requires one block erasure operation before the block can be rewritten. An SSD's endurance depends on many internal (often proprietary) parameters, some of which are hard or impossible to measure: internal write-amplification factor, wear-leveling techniques, FTL's effectiveness, garbage collection algorithms, reserved space, internal caching, and more. In this paper, we do not attempt to measure these internals. Instead, to help estimate the SSD's endurance, we used 4KB as the default SSD page size, counted each page access (read and write) to the SSD, and formalized our endurance model to study the trade-offs among performance, energy, and endurance.

Moreover, as the real-time endurance relies heavily on the history usage of the devices, GreenDM utilizes delta endurance metrics for both SSD and HDD to show the endurance reduction of each device in any configured experiment. We summarize the endurance models as follows:

$$Endu_{ssd}(t) = 1 - \frac{writes(t)}{Limit_{ssd}}(t > 0) \tag{6.4}$$

$$Endu_{hdd}(t) = 1 - \frac{\#startstop(t)}{Limit_{hdd}}(t > 0) \tag{6.5}$$

$$\Delta Endu_{ssd}(\Delta t) = \frac{\Delta writes}{Limit_{ssd}}(\Delta t > 0) \tag{6.6}$$

$$\Delta Endu_{hdd}(\Delta t) = \frac{\Delta \#startstop}{Limit_{hdd}}(\Delta t > 0) \tag{6.7}$$

$$0 \leq Endu_{ssd}(t), Endu_{hdd}(t) \leq 1(t > 0) \tag{6.8}$$

$Endu_{ssd}(t)$ and $Endu_{hdd}(t)$ represent the endurance metric of the SSD device and the HDD device, at time t, respectively. $\Delta Endu_{ssd}(\Delta t)$ and $\Delta Endu_{hdd}(\Delta t)$ represent the delta endurance (i.e., the endurance reduction) of the SSD device and the HDD device during the time period Δt, respectively. The endurance of SSD at time t is represented by 1 minus the fraction of writes performed at time t (i.e., $writes(t)$) and the total writes limit (i.e., $Limit_{ssd}$). The more writes are performed, the less durable the SSD is. Note that reads also affect the SSD's endurance because erase operation will be incurred once read disturbance correction kicks in [84]. Since this is fairly recent reported result and there is no quantitative study on the endurance effects of the read disturbance, in our work, we convert the effect of reads to writes based on several certain ratios (e.g., endurance effects caused by reads is calculated by $reads/10$ and $reads/100$). The endurance of the HDD at time t is represented by 1 minus the fraction of start-stop cycles performed at time t (i.e., $\#startstop(t)$) and the total cycles limit (i.e., $Limit_{hdd}$). The more the device performs start-stop actions, the less durable the HDD is, and the closer it is to failing. We show the limits for both SSD and HDD in table 6.2 based on the vendor data-sheet.

To simplify the understanding and use of our endurance metric, we define eu as the unit for the endurance models as shown in Equation 6.8. We define endurance on a scale of one million parts. The higher the value is, the more durable the device is: a value of 1,000,000 is given to brand new drive that is unlikely to break under failure mode I, and a value of 0 is given to a drive that is almost certain to break under failure mode I in the very near term. For example, a reduction of a device's endurance by $1,000eu$ means that the probability of a device's failure has increased by $\frac{1{,}000}{1{,}000{,}000}$ or 0.1%.

6.1.6 Implementation Details

Concurrency control The Linux DM framework supports concurrent block accesses. Since data migration is performed in the back-end, it is possible that data migration and an incoming I/O compete for the same extent. GreenDM uses a spin-lock to protect critical resources, and creates one atomic counter for each extent to ensure that before GreenDM migrates data, all I/O requests on associated extents are completed. This counter is incremented once per access on the extent, and is decremented for each I/O request that is finished. If GreenDM observes that the counter of one specific extent is larger than zero, it drops the data migration attempt. If the incoming I/O happens to be in the extent that data migration is going to be performed, GreenDM delays the I/O by putting it into a queue and serve it later.

Metadata management Metadata (e.g., mapping table and bitmap) is critical for a data-migration based approach. GreenDM stores metadata in RAM for frequent accesses. In case of a power outage, the system may be inconsistent and lose persistent data. Therefore, GreenDM periodically flushes metadata to the SSD for recovery. GreenDM also replicates metadata on the HDD for redundancy. In case of failures, GreenDM reads the latest metadata checkpoint from one of the persistent drives into RAM. We discuss limitations of the in-memory metadata management in Section 6.4.

Statistics export To better analyze the dynamic mappings of block I/Os, the effectiveness of data migrations, and the status of the running system, GreenDM exports several kernel-space statistics

to user space. GreenDM creates a debugfs entry named "stats" to collect statistic information of the running system (e.g., the SSD hit ratio, the number of promotions and demotions, the system status, etc). GreenDM creates a debugfs entry named "table" to export the mapping table to user level. These statistics were helpful during the development and analysis phases.

Development cost We spent 2 years on this project. We developed around 3,500 LoC in kernel space for GreenDM, and developed fewer than 100 LoC to plug-in statistic code for Linear. We used Auto-pilot [146] to help benchmarking, but further developed an additional 2,000 LoC in Bash and Python to assist in benchmarking and analysis. To automate raw data parsing and plotting, we developed another 2,000 LoC in Bash and Python. To help profiling the performance, we developed another 1,000 LoC in Python. We also developed around 500 LoC in C++ to replay the traces.

6.2 Evaluation

1. What are the GreenDM performance, energy, power, and endurance results compared with other baselines under various workloads?

2. What are the trade-offs among performance, energy, power, and endurance?

3. How much do different tunable parameters affect the trade-offs among performance, energy, and endurance under various workloads?

6.2.1 Experimental Setup

We experimented on two identical Lenovo® ThinkCenter computers. Using lmbench [17], we verified that the performance difference of the two machines was within 2%; and that the power consumption difference was within 1.6%. Each server has 4GB RAM and one Intel® Core™ 2 Quad 2.66GHz CPU. We configured the BIOS identically on both machines. As energy consumption is important to our study, we used the default "ondemand" CPUFREQ governor [156] and the default "menu" CPUIDLE governor [75]. We kept all CPU cores online by default. Our tiering hybrid drive consists of an Intel SSDSA2CW300G3 300GB SSD and a Seagate ST32000641AS 2TB HDD. We used only the middle portion of the HDD's LBA space to average out any ZCAV [151] effects. The OS, using a Linux 3.5.0 kernel, ran on a separate SATA drive. We prepared several baselines: (1) SSD-only drive; (2) HDD-only drive; and (3) a linear tiering hybrid drive (i.e., *Linear*) that linearly maps from the VBA space to the LBA space. We added a few statistics counting code to Linear and named it *Mylinear* in our experiments. We set the DM `split_io` option so that I/Os are split based on the Size (ES). We used $1/4$ as the default ratio for the SSD partition size out of the total drive size for our hybrid drive. The ratio is just one example for us to study the trade-offs among performance, energy, and endurance for the hybrid drive, and it also keeps the SSD size relatively small compared with the workload working set size.

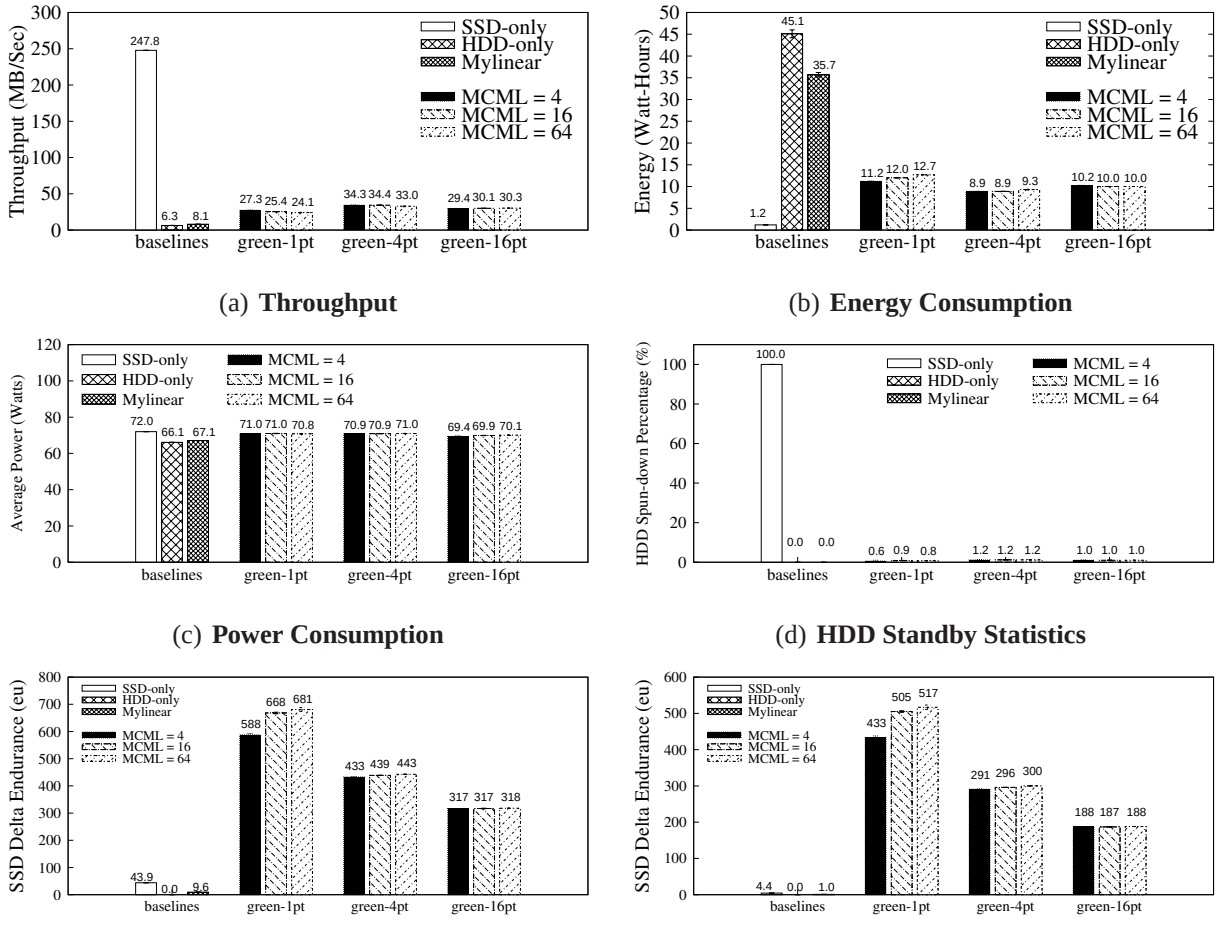
We connected each computer to a WattsUP Pro ES in-line power meter [136], which measures the energy drawn by a device plugged into the meter's receptacle. The power meter uses non-volatile memory to store measurements every second. Its resolution is 0.1 Watt-hours (1 Watt-hour = 3,600 Joules) for energy measurements. The accuracy is $\pm 1.5\%$ of the measured value plus a constant error of ± 0.3 Watt-hours. Its resolution for power measurements is 0.1 Watts. We used

Workload	Drive Size	Reads		Writes	
		Total	Avg Sz	Total	Avg Sz
Web-search	32GB	1,055,236	16KB	212	8KB
FIU online	8GB	655,526	8KB	4,211,786	4KB

Table 6.3: **Trace Workloads Summary**

the `wattsup` Linux utility to download the recorded data from the meter over a USB interface to the test machine.

6.2.2 Benchmarks



(a) **Throughput**

(b) **Energy Consumption**

(c) **Power Consumption**

(d) **HDD Standby Statistics**

(e) **SSD Delta Endurance with Ratio** $1/10$

(f) **SSD Delta Endurance with Ratio** $1/100$

Figure 6.5: **Web-Search Trace Replay Results**. We configured ES to 1MB in GreenDM, to help with sequential prefetching. HDD spin-down was enabled for all.

We evaluated GreenDM carefully with three general purpose workloads: (1) Web-search trace workload; (2) FIU's online trace workload; and (3) File-server workload. We used the Web-search

and FIU online trace workloads from the UMass Trace Repository [130] and the FIU Trace Repository [36], respectively. We summarized these traces' parameters in Table 6.3. Note that the drive sizes are sized to meet the storage requirements of the two workloads, respectively. We used the File-server workload from Filebench [35].

GreenDM's effectiveness depends on the amount of data locality the workload exhibits. Therefore, for the File-server workload, we varied the frequency that files are accessed using Filebench's Gamma distribution [142, 145].

In this Chapter, to understand our GreenDM's behavior under different conditions, we focused on parameters that tend to have more impact on the trade-offs among performance, energy, and endurance of the tiering hybrid drive. Thus, for example, we tuned ES, PT, and MCML values for different workloads while keeping the default values for other parameters (i.e., TW=60, LT=64, HT=128). Specifically, we varied: (1) the Promotion Threshold (PT) and the Maximum Concurrent Migration Limit (MCML) values for the Web-search trace workload; (2) the Extent Size (ES) for the FIU online trace workload; and (3) the MCML and the Gamma values for the File-server workload. To reduce side-effects due to the SSD's Garbage Collection (GC), we issued the TRIM [85] command to the SSD before each experiment. The results we show are general to illustrate the effects of tuning parameters.

We ran all tests a minimum of three times unless otherwise noted. We computed the standard deviations and presented as error bars in figures. We used Autopilot [146] to automate the benchmarks.

6.2.3 Web-Search Trace Workload

We replayed the UMass Web-search trace with our own tool in synchronous mode, without introducing any delay between two consecutive I/O requests. Since the trace is block-level, we disabled the OS buffer cache in this experiment. To meet the storage requirement (i.e., 32GB), we set up the tiering hybrids with the first 8GB from the SSD and the remaining 24GB from the middle of the HDD. Note that the ratio between the SSD and the HDD was not chosen according to the full capacity of the two devices we used in the experiments, but was rather chosen based on the total workload's working set size so that the SSD capacity is kept relatively small compared to the total workload's working set size. We scanned the device initially to fill the mapping table such that it could represent a more realistic situation where the mapping table was not initially empty. We present the results in Figure 6.5.

Figures 6.5(a), 6.5(b), and 6.5(c) show that: (1) the SSD-only drive achieves the highest throughput, the lowest energy consumption, and the highest power consumption; (2) the HDD-only drive achieves the lowest throughput, the highest energy consumption, and the lowest power consumption; (3) tiering hybrids achieve throughput, energy and power consumption in the middle; and (4) among tiering hybrids, various GreenDM configurations achieve higher throughput, lower energy consumption, and higher power consumption compared with Mylinear since the real-world Web-search trace exhibits many hot and cold I/O patterns for GreenDM to manage. Figure 6.5(d) shows that the HDD is rarely spun down for all benchmarks when the disk spin-down in enabled. The reason is that this workload exhibits high randomness and therefore keeps the HDD active most of the time. Thus, the incurred reduction to the HDD's endurance can be ignored. Figures 6.5(e) and 6.5(f) show that: (1) the HDD-only drive does not reduce the SSD's endurance since there are no I/O accesses to the SSD; (2) the Mylinear tiering hybrid drive wears out the SSD the slowest

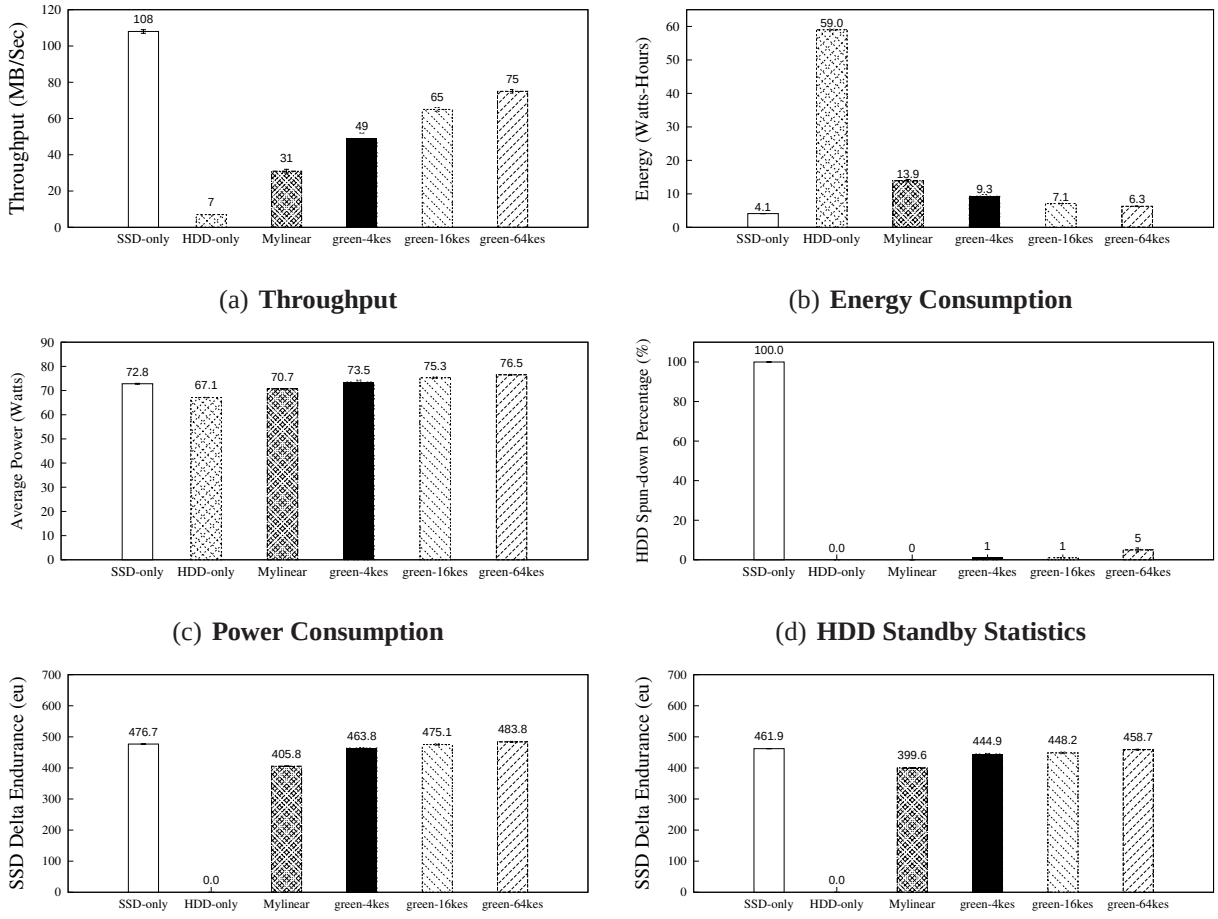
since part of the I/Os goes to the HDD; (3) the GreenDM tiering hybrid drive wears out the SSD the fastest since data migration causes lots of SSD reads and writes; and (4) the SSD-only drive reduces the SSD endurance in the middle since there is no data migration at all. As tiering hybrids can better trade-off performance, capacity, and cost, we focus our study on the trade-offs for tiering hybrid drives.

Higher throughputs lead to larger energy savings, as shown in Figures 6.5(a) and 6.5(b). The reason is that it takes less time to finish the same amount of work when the throughput is higher and the system-level average power consumption between GreenDM and Mylinear is close (see Figure 6.5(c)).

There are trade-offs between performance and power consumption. As shown in Figures 6.5(a) and 6.5(c), GreenDM achieves higher throughput (198–325%) than Mylinear, but consumes slightly more power (5%) since the faster SSD I/Os indirectly keep the CPU and RAM busier, and shift the bottleneck a bit towards the CPU. This keeps the system more active during the run, and shows the trade-off relationship between performance and power consumption for this workload. Note that the SSD-only based system consumes a little bit higher power (1%) than GreenDM because it makes the CPU and RAM even busier.

There are trade-offs between performance and the SSD endurance. As shown in Figures 6.5(a), 6.5(e), and 6.5(f), GreenDM achieves higher performance than Mylinear, but reduces the SSD's endurance more. When the ratio of read-to-write effect is $1/10$, the reduction goes from $32\times$ to $70\times$ more. When the ratio of read-to-write effect is $1/100$, the reduction goes from $186\times$ to $516\times$ more. The reason is two-fold: (1) GreenDM performs many data migrations to separate hot and cold data; and (2) Web-search workload has many more reads than writes and reads are not as effective as writes in reducing the SSD's endurance [84]. Note that MCML values become less effective when the PT value becomes larger since a larger PT value leads to smaller promotions.

Different GreenDM tunable parameters have different effects on performance, energy, and device endurance. As shown in Figures 6.5(a), 6.5(b), 6.5(d), 6.5(e), and 6.5(f), different MCML and PT values affects the performance, energy, and endurance in different ways. For example, when the PT and MCML values are 1 and 64, respectively, GreenDM improves throughput by 198%, saves energy by 64%, and reduces the SSD's endurance by $70\times$ and $516\times$ more when the ratio of read-to-write effect is $1/10$ and $1/100$, respectively. However, when the PT and MCML values are 4 and 16, respectively, GreenDM improves throughput by 325%, saves energy by 75%, and reduces the SSD's endurance by $44\times$ and $295\times$ more when the ratio of read-to-write effect is $1/10$ and $1/100$, respectively. The reason is that different GreenDM parameters yield different benefits and (CPU and I/O) overhead. Medium PT and MCML values tend to achieve the best balance of benefits vs. overhead (see Figure 6.5(a)) for this workload: (1) a too large PT value can reduce the migration benefits and a too small PT value can increase the migration overhead for this workload; (2) when the PT value is small, large MCML values can consume more CPU and I/O resources on the system; and (3) when the PT value is large, large MCML values can promote hot I/Os to the SSD faster. However, when the PT value is larger, it incurs less reduction to the SSD's endurance (see Figures 6.5(e) and 6.5(f)). In sum, there is no single best configuration for this workload. Therefore, to achieve different trade-off goals, the MCML and PT values have to be chosen carefully.



(a) **Throughput**

(b) **Energy Consumption**

(c) **Power Consumption**

(d) **HDD Standby Statistics**

(e) **SSD Delta Endurance with Ratio** $1/10$

(f) **SSD Delta Endurance with Ratio** $1/100$

Figure 6.6: **Online Trace Replay Results**. As example, we set MCML to 16 and PT to 1. Disk spin-down was enabled for all.

6.2.4 FIU Online Trace Workload

We replayed the FIU online trace using our own tool as mentioned in Section 6.2.3. We disabled the OS buffer cache as the trace is a block-level one. To meet the storage requirement (i.e., 8GB), we set up the tiering hybrids with the first 2GB from the SSD and the remaining 6GB from the middle of the HDD. We scanned the device initially to fill the mapping table. We present the results in Figure 6.6.

Figures 6.6(a), 6.6(b), and 6.6(c) show that: (1) the SSD-only drive achieves the highest throughput, the lowest energy consumption, and a medium power consumption; (2) the HDD-only drive achieves the lowest throughput, the highest energy consumption, and the lowest power consumption; (3) tiering hybrids achieve throughput and energy consumption in the middle; and (4) among tiering hybrids, various GreenDM configurations achieve higher throughput, lower energy consumption, and higher power consumption compared with Mylinear, due to GreenDM's efficient data management. Figure 6.6(d) shows that GreenDM spins down the HDD to some degree when

the ES varies. Otherwise, the HDD is rarely spun down for other benchmarks. Since only a few start-stop cycles are caused, the reduction to the HDD's endurance can be ignored. Keeping the HDD idle could save some power, but since the SSD indirectly helps the CPU stay busier and the spin down/up process consumes more power, the GreenDM system-level power consumption is thus slightly higher than all others. Figures 6.6(e) and 6.6(f) show that: (1) the HDD-only drive does not reduce the SSD's endurance; (2) the SSD-only drive wears out the SSD to a moderate degree compared with tiering hybrids; and (3) GreenDM configurations wears out the SSD faster than Mylinear. Next we discuss the trade-offs for tiering hybrid drives.

Higher throughputs lead to lower energy consumption, as shown in Figures 6.6(a) and 6.6(b). The reason is similar to what we have explained in Section 6.2.3.

There are trade-offs between performance and power consumption. As shown in Figure 6.6(c), GreenDM achieves higher throughput (58–142%) than Mylinear. However, it consumes more system-level power on average than Mylinear does, ranging from 4–8%, due to the aforementioned reasons.

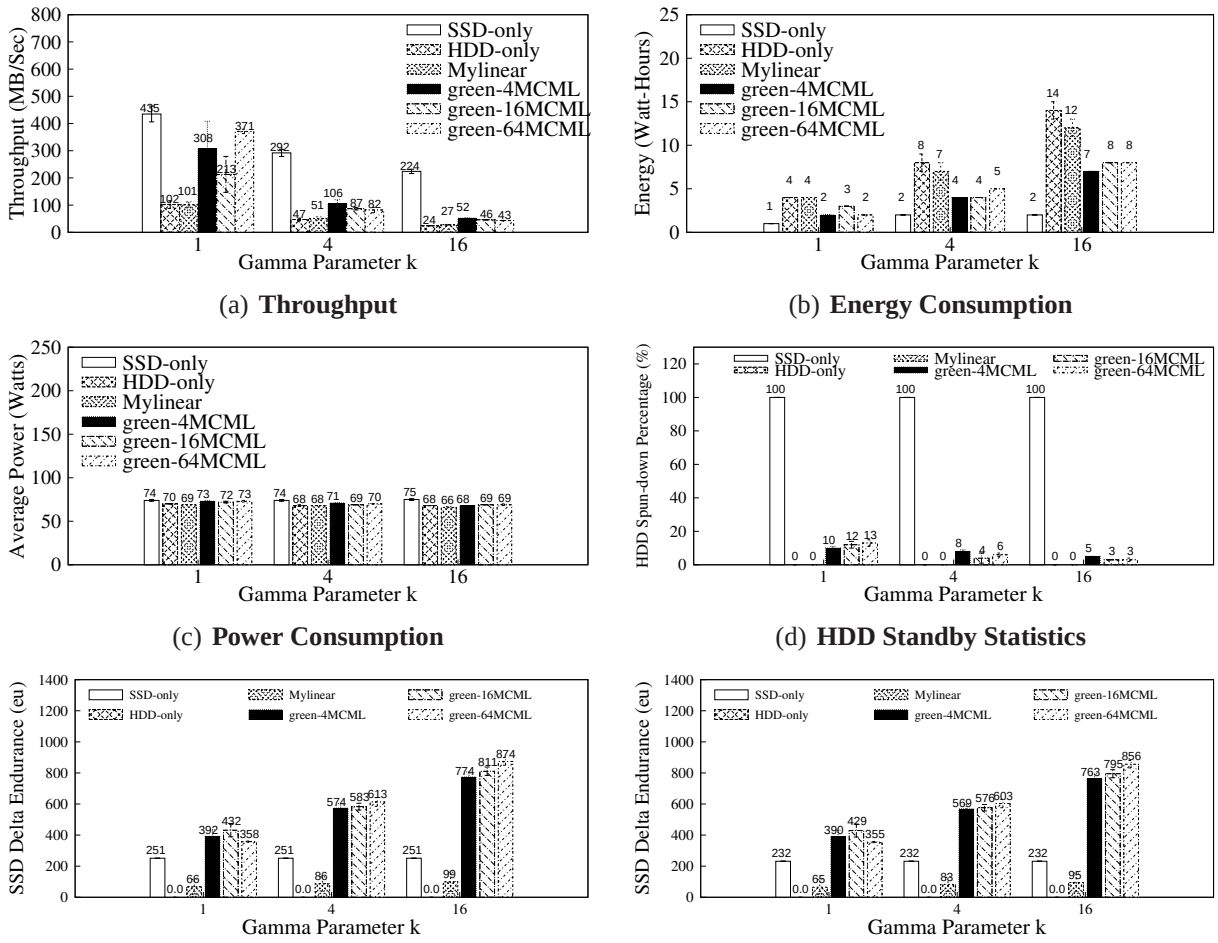
There are trade-offs between performance and the SSD endurance. As shown in Figures 6.6(a), 6.6(e), and 6.6(f), GreenDM achieves higher throughput than Mylinear does, but it reduces the SSD's endurance by 14–19% more and by 11–15% more when the ratio of read-to-write effect is $1/10$ and $1/100$, respectively, as we explained in Section 6.2.3.

Different GreenDM Extent Sizes (ES) have different effects on GreenDM's performance, energy consumption, and device endurance. As shown in Figures 6.6(a), 6.6(b), 6.5(d), 6.6(e), and 6.6(f), different ES values lead to different results. For example, when the ES is 4KB, GreenDM improves throughput by 58%, saves energy by 33%, and reduces the SSD's endurance by 14% and 11% more when the ratio of read-to-write effect is $1/10$ and $1/100$, respectively. However, when the ES is 64KB, GreenDM improves throughput by 142%, saves energy by 55%, and reduces the SSD's endurance by 19% and 15% more when the ratio of read-to-write effect is $1/10$ and $1/100$, respectively. The larger the ES is, the more effective the sequential pre-fetching algorithm is. Therefore, it leads to higher throughput and larger energy savings. However, larger ES causes more reduction to the SSD's endurance. It suggests the ES has to be chosen carefully for the system to achieve the best trade-offs because there is no single optimal configuration.

6.2.5 File-Server Workload

We ran the File-server workload with a Gamma distribution in Filebench [35]. We varied the Gamma value to show different results. The smaller the Gamma value is, the higher the data locality is since smaller Gamma values lead to narrower file accesses: that is, a certain subset of data items (i.e., Logical Blocks) would be referenced more than others. We configured the usable RAM size to be 1GB to ensure that the workload would generate many low-level I/Os. To meet the storage requirement (i.e., 8GB), we set up the tiering hybrids with the first 2GB from the SSD and the remaining 6GB from the middle of the HDD. We report the results in Figure 6.7. Note that since the OS buffer cache is enabled, to better estimate the SSD endurance change, we assume all I/Os for SSD-only benchmark go to the low-level device.

Figures 6.7(a), 6.7(b), and 6.7(c) show that: (1) the SSD-only drive achieves the highest throughput, the lowest energy consumption, and the highest power consumption; (2) the HDD-only drive achieves the lowest throughput, the highest energy consumption, and medium power consumption; (3) tiering hybrids achieve throughput, energy and power consumption in the middle



(a) **Throughput**

(b) **Energy Consumption**

(c) **Power Consumption**

(d) **HDD Standby Statistics**

(e) **SSD Delta Endurance with Ratio** $1/10$

(f) **SSD Delta Endurance with Ratio** $1/100$

Figure 6.7: **Fileserver Workload Results**. We configured the ES to be 128KB in GreenDM. It is equal to the average I/O size to avoid the migration waste and I/O split overhead. The PT value was fixed at 1, as example. Disk spin-down was enabled for all.

in general; and (4) among tiering hybrids, various GreenDM configurations achieve higher throughput, lower energy consumption, and higher power consumption compared with Mylinear through efficient data management. Note that in Figure 6.7(a), there are larger throughput variations when the gamma parameter is 1. The reason is that when the data locality is high, the OS buffer cache can kick in and make the overall throughput vary wildly, resulting in a bi-modal distribution [65, 126]. We have rerun this experiments ten times more, plotted a histogram, and verified that there were two throughput modes: one from the RAM buffer cache and a second from the low-level tiering hybrid drive. Figure 6.7(d) shows that the HDD is spun down around 10% of the time when Gamma is small. The reason is that when the Logical Blocks (LBs) are more narrowly localized, GreenDM has a larger chance to spin down the HDD. Note that, although GreenDM shows the HDD being spun down, it incurs only a small number of HDD start-stop cycles that can be ignored towards the HDD's endurance reduction. Figures 6.7(e) and 6.7(f) show that: (1) the HDD-only drive does not reduce the SSD's endurance; (2) the SSD-only drive wears out the SSD to a moderate degree com-

pared with tiering hybrids; (3) GreenDM configurations wear out the SSD faster than Mylinear; and (4) larger Gamma value also tends to wear out the SSD faster. Next we discuss the trade-offs on tiering hybrid drives.

Higher throughput leads to larger energy savings. We can see from Figures 6.7(a) and 6.7(b) that when the throughput is higher, the corresponding energy savings are larger under any condition. The main reasons are that: (1) it takes less time to finish the same amount of work when the throughput is higher; and (2) the system-level average power consumption between GreenDM and Mylinear is close (see Figure 6.7(c)).

There are trade-offs between performance and power consumption in GreenDM. As shown in Figures 6.7(a) and 6.7(c), GreenDM achieves higher throughput (50–267%) compared to Mylinear, but consumes 3% more power. The reason is that because the SSD's I/Os are faster than the HDD, the bottleneck is shifted further to the CPU and RAM, making the whole system more active and consuming more power, even though the HDD is spun down in some degree.

There are trade-offs between performance and SSD endurance. As shown in Figures 6.7(e) and 6.7(f), GreenDM achieves higher throughput than Mylinear does, but it wears out the SSD faster from 4× to 8×, because of the same reason as we explained in Section 6.2.3. Moreover, a larger Gamma value can wear out the SSD faster. The reason is that when the Gamma parameter is larger, I/Os are distributed over a wider range of LBs. Hence, there are more promotions and demotions, which eventually increases the SSD read and write counts and reduces the SSD's endurance.

Different GreenDM tunable parameters have different effects on performance, energy, and device endurance. As shown in Figures 6.7(a), 6.7(b), 6.7(d), 6.7(e), and 6.7(f), different MCML values under different data locality affect the performance, energy, and endurance in different ways. For example, when the Gamma and MCML values are 1 and 64, respectively, GreenDM improves throughput by 267%, saves energy by 50%, and reduces the SSD's endurance by 4× more. However, when the Gamma and MCML values are 16 and 4, respectively, GreenDM improves throughput by 93%, saves energy by 71%, and reduces the SSD's endurance by 7× more. The reason is that different GreenDM parameters create different benefits and (CPU and I/O) overhead under different data locality: (1) when the Gamma value is small, larger MCML values can promote hot data to the SSD faster; and (2) when the Gamma value is large, smaller MCML values incurs less CPU and I/O overhead. However, different configurations with different Gamma values wear out the SSD to a different degree. Therefore, to meet different requirements, tunable parameters have to be chosen carefully.

6.2.6 Summary

In this section, we summarize the best configuration for each of workloads under the current setup. Future storage system designs can potentially refer to the conclusion here.

1. For the read-intensive Web-search workload, medium PT and MCML values lead to the best throughput for GreenDM due to the net effect of migration benefit over overhead. In terms of SSD endurance reduction, the PT value is the dominant factor since it affects the SSD access to a large degree. A smaller PT value wears out the SSD faster since it incurs more SSD accesses.

2. For the write-intensive Online workload, a larger ES value leads to higher throughput for

GreenDM since it introduces more efficient prefetching. However, a larger ES value wears out the SSD faster since it incurs more SSD accesses.

3. For the read- and write- intensive fileserver workload, the Gamma value is the dominant factor for GreenDM's throughput and SSD endurance reduction since data locality matters significantly for GreenDM to work. When Gamma is small, a large MCML value achieves better throughput and wears out the SSD slower since GreenDM migrates hot I/Os earlier for both GreenDM and the OS buffer cache. When Gamma is large, a small MCML value achieves better throughput and wears out the SSD slower since it incurs less overhead for GreenDM and the OS buffer cache.

4. For all workloads, since the system (e.g., CPU and RAM) becomes busier due to GreenDM's data management, the system consumes slightly more power than the Mylinear baseline does.

5. For all workloads, since the system power consumption is not largely different, a larger throughput leads to smaller energy consumption when the total amount of workload is fixed.

6.3 Related Work

Our work is different from past ones in three ways: (1) we used a tiering hybrid drive; (2) we developed an endurance metric and studied the trade-offs among performance, energy, and device endurance in a tiering hybrid drive empirically; and (3) we offered a versatile solution to enable the system parameters to be tuned for specific workloads.

There are many existing systems exploring SSD as a cache [4, 11, 37, 53, 70, 73, 83, 92–94, 103, 112, 129, 133], where SSDs are used to cache data. Only some [23, 49, 68, 123, 129, 147] have explored using SSDs as primary storage to better trade-off throughput, capacity, and cost. There are also several tiering hybrid drives in industry: Apple's Fusion Drive [141], Microsoft's Ready Drive [100], Western Digital's Solid State Hybrid Drive (SSHD) [138], Tintri's VMstore [134], and Dell's Compellent Flash Array [30]. However, most of their internal designs and source code are not publicly available. GreenDM is a tiering hybrid drive, whose source code and internal designs are scheduled to be released under the GPL for the entire community to utilize.

Many performance, energy, and endurance relevant studies are simulated: FlashTier [112], SieveStore [103], BEST [53], HybridStore [68], Pearl [147], GreenHDFS [67], PDC [101], NVCache [14], and FAWN [5]. While simulation can help provide early useful results, we believe that empirical experiments are more realistic. GreenDM performs real-world experiments to study the trade-offs among performance, energy, and endurance of a tiering hybrid drive.

For the SSD endurance metric, past studies normally refer to the number of erasure cycles that can be performed on an SSD during its lifetime [74], and do not provide a concrete model and metric to help study the SSD's endurance. One study [131] explores a hardware-specific SSD endurance model. While it is useful in some cases, it requires hardware parameters (e.g., voltage, density, etc.) to estimate the endurance through simulation, and can be inconvenient for user-level endurance estimation in reality. GreenDM goes further by developing an endurance model and metric to help study the trade-offs among performance, energy, and endurance in a versatile hybrid drive.

Many storage systems use SSDs as the front tier, but they aim for high performance, efficient energy consumption, or improved endurance. Thus, they often do not closely study the trade-offs among performance, energy, and endurance. Moreover, many studies do not offer flexible policies to enable adaptation to different workloads. HybridStore [68] consolidates SSDs and HDDs for a cost-efficient storage system while meeting the performance and lifetime requirements. It is based on simulation, and does not study the trade-offs among performance, energy, and endurance of the tiering hybrid drive. EDT [49] dynamically migrates a fixed-size extent among different tiers to satisfy performance requirements and reduce power consumption. BTIER [129] uses a fast storage medium for caching and migrates aged data to a lower tier over time for high performance. Its migration policies are somewhat configurable, but it does not consider the power consumption and the endurance of the tiered storage. Pearl [147] tries to balance the performance, energy, and reliability of disk arrays by migration. It relies on simulations alone and does not empirically study the trade-offs in details. Hystor [23] and Aggregate [123] use SSDs as the front tier primary storage for high performance only. GreenHDFS [67] explores how to divide servers in a data-center into different zones to save power while maintaining performance. PDC [101] discusses how to migrate data center workloads to fewer disks so that others can be put into lower-power states to save energy. NVCache [14] utilizes NVRAM for the I/O subsystem for lower power consumption. GreenFS [64] allows hard disks to be kept off most of time to minimize the disk-drive-related power consumption. MAID [24] uses data placement, scrubbing, and recovery techniques to put many of the drives in the system into a low-power mode to save energy. Pergamum [122] adds NVRAM at each storage node to allow inter-disk data verification while the disk is powered off to save power in a distributed system. PARAID [139] allows adaptive transitions between several different RAID layouts to trade off energy, performance, and reliability. FAWN [5] uses "wimpy" nodes with power-efficient CPUs and I/O capabilities to save power while achieving performance and scalability in a distributed system.

GreenDM is different from the above approaches. It explores in depth the trade-offs among performance, energy, and device endurance in a hybrid drive and comes with a versatile approach so that important system parameters can be investigated and traded off to be best tuned for specific workloads.

6.4 Limitations

While GreenDM estimates the endurance metric by counting the SSD reads and writes and the start-stop (spin-up/down) cycles of the HDD, the endurance metric can be improved. A finer-grained counting in terms of the internal SSD erasure cycles and the FTL's behavior could help build a more accurate endurance estimation for the SSD.

GreenDM provides coarse-grained control (i.e., tunable parameters) to trade-off performance, energy, and device endurance under different workloads. We do not offer fine-grained control (e.g, QoS). To reach that goal, we first have to formally study the relationship between performance, energy, device endurance, and various controllable system parameters. We believe machine-learning-based approaches (e.g., hill-climbing [76] and control theory [78, 155]) could help explore such relationships.

GreenDM currently flushes dirty data periodically. It can be dangerous. To provide transaction support, it requires a journaling mechanism for tiering hybrid storage systems.

We currently build the virtual device from two drives only: an SSD and an HDD. We could potentially scale the current setup to multiple drives and more types (e.g., SAS, Shingled, PCM, and NAS) and develop more generalized techniques.

6.5 Conclusion

We designed, built, and evaluated the versatile tiring hybrid drive to study the trade-offs among performance, energy, and endurance. We presented interesting results for various trade-offs observed. For the FIU online trace workload, GreenDM achieved higher throughput (58–142%) than Mylinear, but consumed more power (4–8%) and further reduced the SSD's endurance by 11–15% when the ratio of read-to-write effect is $1/100$. We demonstrated the importance of matching tunable parameters to different workloads to better trade-off performance, energy, and endurance. For the FIU online trace workload, a larger extent size (ES) lead to higher throughput and larger energy savings, but also further reduced the SSD's endurance.

Next, we discuss the cost evaluation results for GreenDM. It can help justify the performance gained.

Chapter 7

Cost Evaluation

Storage systems are getting more complex with solid-state technologies rapidly taking hold, shingled devices available, and hybrids thereof being proposed and commercialized [94, 133, 134]. As the amount of digital data grows rapidly, virtualization and cloud technologies highlight the need to consolidate storage and save on the longer-term costs of data storage. Complex workloads play a key role in how storage systems behave. The interplay of hardware, software, and workloads can have significant impact on throughout, energy consumption, and device durability. We propose to evaluate modern storage systems from a monetary cost perspective that includes many dimensions as well as traditional performance [42]. We assume that server-class storage systems should be utilized at high yields, due to consolidation and virtualization. We further propose that monetary costs should be evaluated over the expected lifetime of the storage system, typically years, and consider device wear-out and replacement [107].

Several studies integrate SSDs into storage systems, and some consider the original purchase cost or short-term energy use, but neglect to consider the long term impact on device wear-out [49, 52, 68, 92, 103, 123]. Some simulated the results instead of conducting empirical studies [68, 103].

In this chapter, we propose a general cost model to study the cost dimension of storage systems. Our cost model considers not only the initial purchase cost, but also the total cost of ownership over time. Our cost model includes several factors for the total cost of ownership: (1) energy cost; (2) power cost; (3) device endurance cost; and (4) service cost. We also scale the experiments to observe long-term effects.

We conducted extensive experiments using many workloads and configurations—including single-drives and hybrids. We observed that for some workloads, an SSD-only solution incurs the highest overall costs in the short term but much lower costs in the long run. We also observed that hybrids incurs medium initial purchase investments, but can incur long-term costs of varying degrees depending on the workloads. That is why we believe that future storage systems must be evaluated across dimensions of lifetime cost, performance, as well as workloads.

We discuss the cost model in Section 7.1. We provide one working example based on the cost model in Section 7.2. We present the associated cost evaluation results of GreenDM under several workloads in Section 7.3. Our results contain several interesting observations. We discuss related work on cost dimension of storage systems in Section 7.4. We discuss the limitations of our current cost model in Section 7.5. We conclude the cost evaluation chapter in Section 7.6.

7.1 Cost Model

This section details our work in building a cost model to further justify the performance trade-offs explored. It can potentially provide valuable hints to the future design of storage systems.

A cost metric is important to justify storage systems' expenditures [49, 98, 122]. With the advent of Flash-based SSD that is more expensive, the cost dimension of storage systems with SSD deployed is becoming more interesting to explore. Generally, the cost in dollars comes from the upfront purchase and the total cost of ownership (TCO [47,127]). More specifically, the cost model combines several factors below.

1. The upfront capacity purchase.
2. The recurring energy and power cost.
3. The device replacement cost.
4. The service cost (e.g., rack space, man power, etc).

We also use a time factor to predict the long-run cost. To explain the model better, we summarize the calculation formula as follows:

$$1 \leq i \leq n \, (n : the\, number\, of\, devices) \tag{7.1}$$

$$1 \leq TimeFactor \, (integer,\, default = 1) \tag{7.2}$$

$$Cost = Purchase + TCO \tag{7.3}$$

$$Purchase = \sum_{i=1}^{n} Cost_{dev_i} \tag{7.4}$$

$$Cost_{dev_i} = Normalized\, Price_{dev_i} \times Capacity_{dev_i} \tag{7.5}$$

$$TCO = TimeFactor \times (Cost_{energy} + Cost_{power} + Cost_{endurance}) + Cost_{service} \tag{7.6}$$

$$Cost_{energy} = Lookup_{LIPA}(Amount_{energy}) \tag{7.7}$$

$$Cost_{power} = Lookup_{LIPA}(Amount_{power}) \tag{7.8}$$

$$Cost_{endurance} = \sum_{i=1}^{n} Cost_{endu_i} \tag{7.9}$$

$$Cost_{endu_i} = Cost_{dev_i} \times \frac{dev_i\, wearout}{Limit_i} \tag{7.10}$$

$$dev_i\, wearout = \begin{cases} writes \text{ if } dev_i = SSD \\ \#startstop \text{ if } dev_i = HDD \end{cases} \tag{7.11}$$

$$Limit_i = \begin{cases} Limit_{writes} \text{ if } dev_i = SSD \\ Limit_{cycles} \text{ if } dev_i = HDD \end{cases} \tag{7.12}$$

$$Cost_{service} = fixed\, estimation \tag{7.13}$$

Equation 7.1 names a variable (i.e., i) for each of the devices. Equation 7.2 specifies the time factor range for future projection. Equation 7.3 shows that the total cost (i.e., $Cost$) depends on

the upfront purchase cost (i.e., $Purchase$) and the Total Cost of Ownership (TCO) (i.e., TCO). Equations 7.4 and 7.5 show that the upfront purchase depends on normalized price of each device (i.e., $Normalized\,Price_{dev_i}$) and the capacity of each device (i.e., $Capacity_{dev_i}$). Note that the normalized price of each device can change over time. In our thesis, we present the results based on the prices the Intel SSD and Seagate HDD we purchased in 2012.

Prices ($ per KWh/KW)	**under 7KW**		**under 145KW**		**over 145KW**	
	Energy	**Power**	**Energy**	**Power**	**Energy**	**Power**
offpeak	0.0863	0	0.0191	0	0.0218	0
peak	0.1052	0	0.0340	48.78	0.0446	28.76
intermediate	0.0863	0	0.0317	5.94	0.0356	8.13

Table 7.1: LIPA energy and power prices for commercial use as of May 2013.

Equation 7.6 shows that the TCO depends on the energy cost (i.e., $Cost_{energy}$), the power cost (i.e., $Cost_{power}$), the endurance cost (i.e., $Cost_{endurance}$), and the service cost (i.e., $Cost_{service}$). We also use a time factor (i.e., $TimeFactor$) to predict future costs associated with the energy, power, and endurance in a longer run (i.e., run the same workload multiple times). Equations 7.7 and 7.8 show that we can get the energy and power cost by looking up the price table (i.e., $Lookup_{LIPA}$) provided by the local electricity authority (i.e., Long Island Power Authority), as shown in Table 7.1. Note that we currently assume: (1) the energy is distributed by $3/8$, $1/4$, and $3/8$ in accordance with offpeak, peak, and intermediate; (2) the power in offpeak, peak, and intermediate is the average power. The energy and power measurement is based on the whole system. We used a simplified method to estimate the energy and power cost. Equation 7.9 show that we can get the total endurance cost by summarizing each device's endurance cost (i.e., $Cost_{endu_i}$). Equation 7.10 shows that we can get each device's endurance cost by multiplying the wear out degree (i.e., $\frac{dev_i\ wearout}{Limit_i}$) of each device type by the device's cost (i.e., $Cost_{dev_i}$). Note that if the device is worn out a certain degree, we then need to save money accordingly to buy a new device. Note that the wear-out degree and the endurance limit of each device may be different.

	Limits
SSD	36,500 GB writes
HDD	300,000 spin up/down cycles

Table 7.2: Devices wear-out limits.

Equations 7.11 and 7.12 show that the Flash-based SSD endurance depends more on the writes wear out (i.e., $writes$). Note that reads also affect SSD's endurance. In our work, we convert the effect of reads to writes based on a certain ratio (i.e., writes caused by reads is calculated as $reads/10$). They also show that for HDD, the number of HDD start-stop cycles (i.e., $\#startstop$) is one major factor. Other factors include vibration, sector errors, and so on [102]. We use the number of HDD start-stop cycles for simplicity. Table 7.2 present the detailed limits from the hardware manufacture. Equation 7.13 shows that we can further use fixed estimation as the service cost ($Cost_{service}$) for the hardware setup. To better understand the model as we described above, we come up with a working example in the following section.

7.2 Working Example

This Section illustrates how the above cost model works using some example numbers.

Suppose, we have a tiering hybrid drive of 32GB capacity in total. The tiering hybrid drive is composed of one Flash-based SSD and one HDD: 8GB from the SSD and 24GB from the HDD. The total capacity of the SSD device is 300GB and the cost for the device is $529. The total capacity of the HDD device is 2TB and the cost of the device is $200. The normalized prices for the SSD and HDD are $1.76 per GB and $0.1 per GB, respectively. The Flash-based SSD can endure a total amount of 36,500GB writes, and the HDD can endure a total amount of 300,000 start-stop cycles. The tiering hybrid drive is installed in one rack server machine that occupies some space that costs $100 one time in total. The server runs some benchmark (e.g., fileserver workload), which consumes energy and power. Suppose the server finishes running 100GB amount of fileserver workload, and consumes a total amount of 8kWh energy and a average power of 100W. Suppose running the workload causes 365GB worth of writes to the SSD and 10 start-stop cycles to the HDD.

Now, what would be the associated cost according to the above model?

The total cost of purchase would be $16.48: $14.08 ($\$1.76 \times 8$) comes from the SSD device cost and $2.4 ($\$0.1 \times 24$) comes from the HDD device cost. Since the average power is less than 7KW, the energy cost would be $0.6419 ($\$0.0863 \times 3 + \$0.1052 \times 2 + \$0.0863 \times 3$). The power cost would be $0 ($\$0 \times 0.1$). The endurance cost for SSD would be $5.29 ($\$529 \times 365/36500$). The endurance cost of the HDD would be $0.01 ($\$200 \times 10/200000$). Since the service fee is $100 and the time factor is 1, the total cost of running the 100GB fileserver workload would be $122.4219 ($\$16.48 + 1 \times (\$0.6419 + \$0 + \$5.29 + \$0.01) + \$100$). If the time factor becomes 100, then the total cost would change to be $710.67 ($\$16.48 + 100 \times (\$0.6419 + \$0 + \$5.29 + \$0.01) + \$100$).

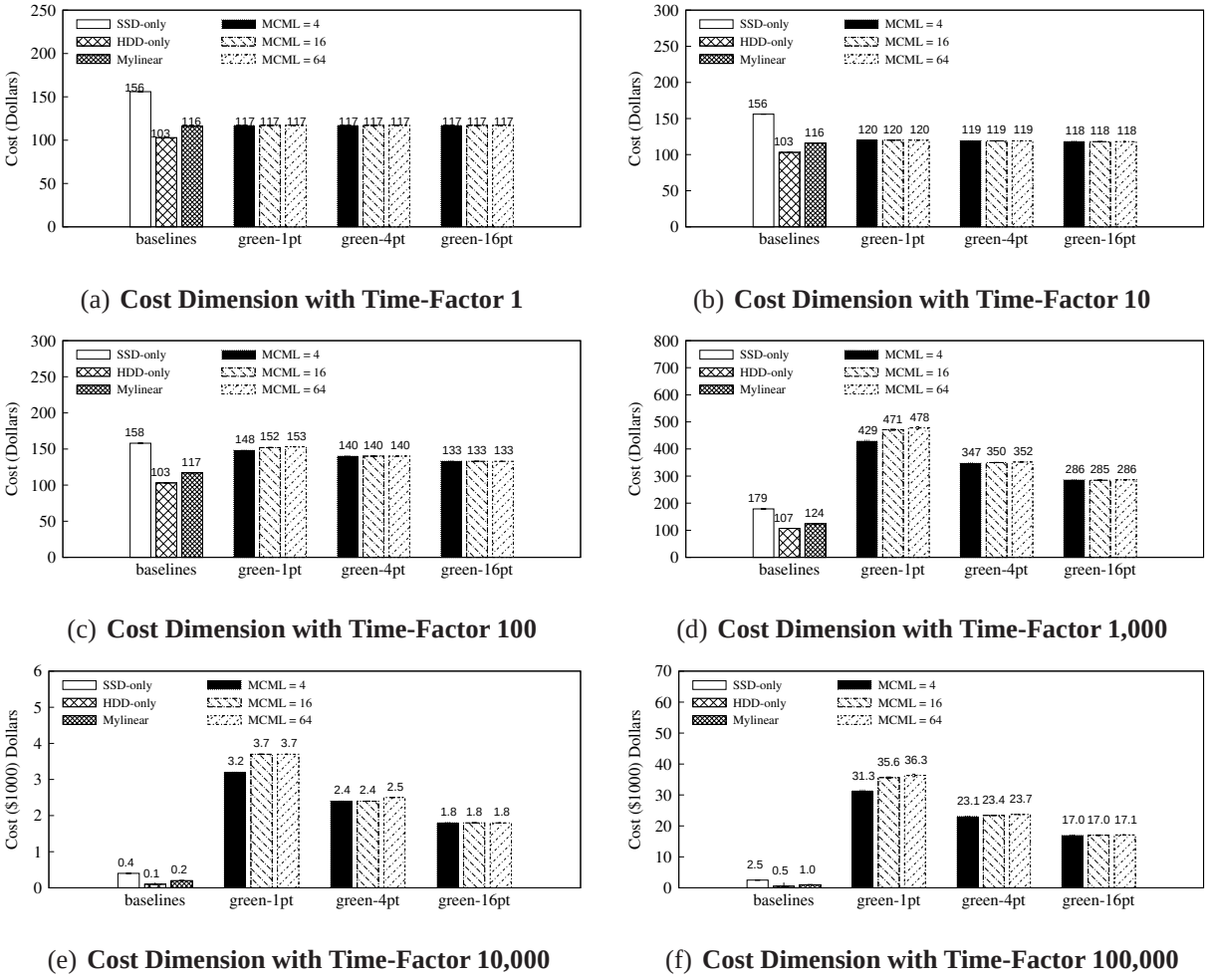
Next, we evaluate the associated costs under several workloads in the following sections.

7.3 Cost Results

In this Section, we present the associated cost results for GreenDM under various workloads as we discussed in Chapter 6.

7.3.1 Web-Search Trace Workload

We present the results of Web-search trace workload in Figure 7.1. As we can see from Figures 7.1(a), 7.1(b), 7.1(c), 7.1(d), 7.1(e), and 7.1(f), when the time factor is 1, the SSD-only drive based system has the highest associated cost, the HDD-only drive based system has the lowest associated cost, and tiering hybrids based systems have medium associated costs. However, as the time factor increases, the associated cost of the HDD-only drive based system stays the least, the associated cost of the SSD-only drive and Mylinear tiering hybrid drive based system increases in the middle, and the associated cost of GreenDM tiering hybrid drive based system increases the most. Among tiering hybrids, GreenDM incurs more cost than Mylinear does. The larger the time factor is, the larger the difference is. When the time factor is 1, the difference is only $1. However, when the time factor is 1,000, the difference is maximally $354, and when the time factor is 100,000, the difference is maximally $35,300. The main reason is that GreenDM comes with many



(a) **Cost Dimension with Time-Factor 1**

(b) **Cost Dimension with Time-Factor 10**

(c) **Cost Dimension with Time-Factor 100**

(d) **Cost Dimension with Time-Factor 1,000**

(e) **Cost Dimension with Time-Factor 10,000**

(f) **Cost Dimension with Time-Factor 100,000**

Figure 7.1: Web-Search workload results. We scale the time factor to show various results.

more data migrations that reduces the SSD's endurance more. Note that for Web-search workload, when the time factor is 100,000, it translates to 2.1 years on average (min and max are 0.2 and 7.7 years, respectively) for all types of benchmarks—a reasonable timeframe for long-time storage.

We can also see from the above figures that different GreenDM MCML and PT values lead to different associated costs. The difference varies when the time factor varies. For example, when the time factor is 1, the difference is almost zero; when the time factor is 10, the difference varies from \$0 to \$2; when the time factor is 100, the difference varies from \$0 to \$20; when the time factor is 1,000, the difference varies from \$1 to \$193; when the time factor is 10,000, the difference varies from \$1,600 to \$3,500; and when the time factor is 100,000, the difference varies from \$16,100 to \$35,300. The main reason is that different GreenDM MCML and PT combinations affect the trade-offs of performance, energy, and endurance in different ways as we have already discussed in Chapter 6.

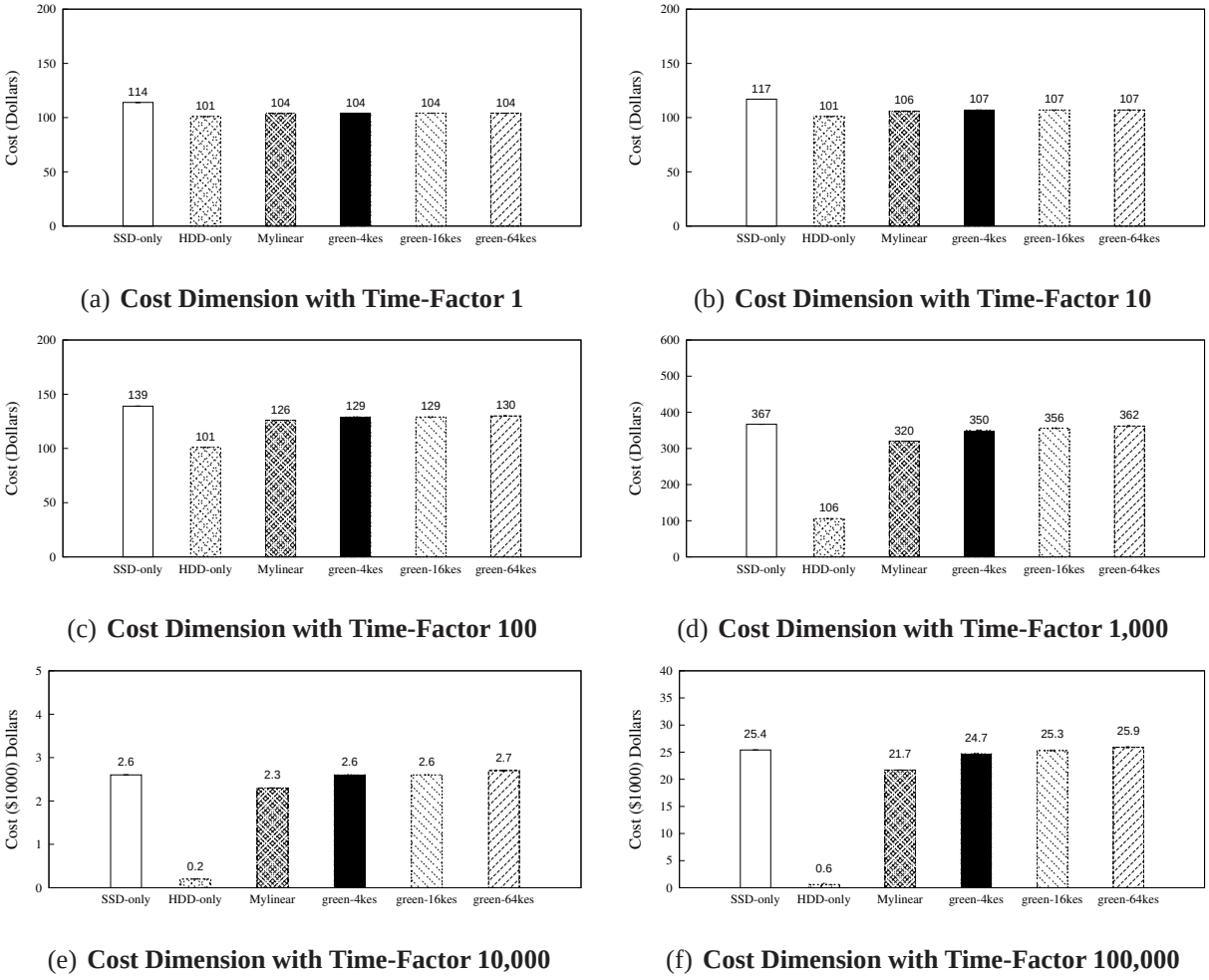
For this workload, a larger PT value tends to incur less cost in the long run since a larger PT causes less number of data migrations. When PT is 1, a smaller MCML value tends to incur less cost in the long run since a smaller MCML value in this case causes less frequent data migration. Therefore, system parameters have to be chosen carefully to justify the gained performance.

7.3.2 FIU Online Trace Workload

We present the results of FIU online trace workload in Figure 7.2. As we can see from Figures 7.2(a), 7.2(b), 7.2(c), 7.2(d), 7.2(e), and 7.2(f), when the time factor is 1, the SSD-only drive based system incurs the highest cost, the HDD-only drive based system incurs the least cost, tiering hybrids based systems incur cost in the middle. However, as the time factor increases, the incurred cost of the HDD-only drive based system increases the least and becomes the smallest one, the incurred cost of the SSD-only drive based system increases greatly and becomes one of the largest ones, and the incurred costs of tiering based systems increase in large degrees as well. Among the tiering hybrids, GreenDM causes more cost than Mylinear does. The larger the time factor is, the larger the difference is. When the time factor is 1, the difference is \$0. However, when the time factor is 1,000, the difference is maximally \$42, and when the time factor is 100,000, the difference is maximally \$4,200. The reason is similar to what we mentioned above. Note that for FIU online workload, when the time factor is 100,000, it translates to 3.3 years on average for all types of benchmarks (min and max are 0.7 and 9.8 years, respectively).

We can also see from the above figures that different GreenDM ES values incur different costs. The difference varies as the time factor varies. For example, when the time factor is 1 and 10, the difference is zero; when the time factor is 100, the difference varies from zero to \$1; when the time factor is 1,000, the difference varies from \$6 to \$12; when the time factor is 10,000, the difference varies from \$0 to \$100; and when the time factor is 100,000, the difference varies from \$600 to \$1,200. The main reason is that different GreenDM ES values also affect the trade-offs of performance, energy, and endurance in different ways as we have discussed in Chapter 6.

For this workload, a smaller ES seems to incur less cost in the long run since smaller ES leads to smaller SSD accesses. Therefore, the ES parameter has to be chosen carefully to justify the gained performance.



(a) **Cost Dimension with Time-Factor 1**

(b) **Cost Dimension with Time-Factor 10**

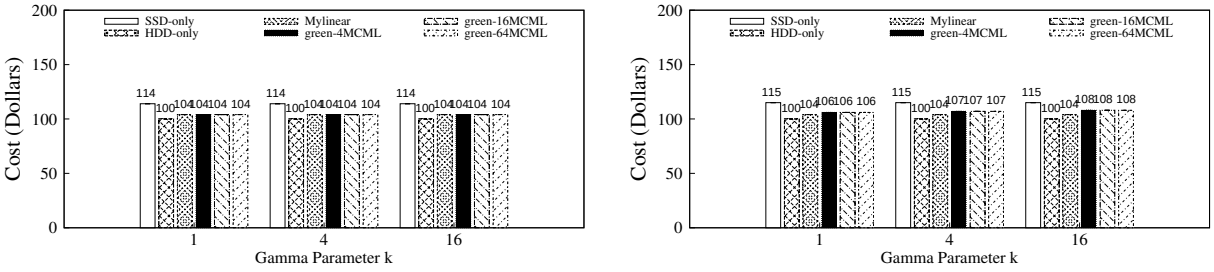
(c) **Cost Dimension with Time-Factor 100**

(d) **Cost Dimension with Time-Factor 1,000**

(e) **Cost Dimension with Time-Factor 10,000**

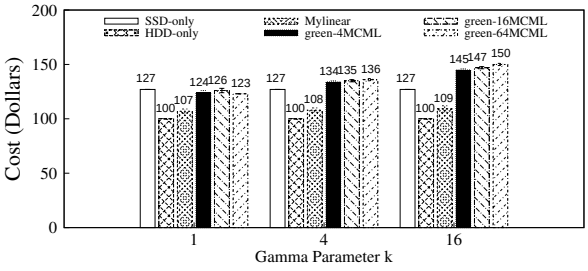
(f) **Cost Dimension with Time-Factor 100,000**

Figure 7.2: Online trace workload results. We scale the time factor to show various results.

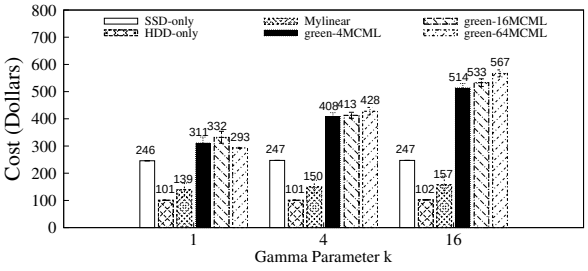


(a) **Cost Dimension with Time-Factor 1**

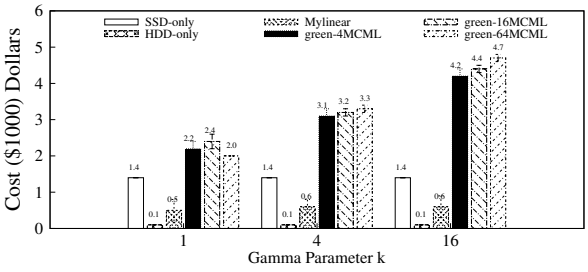
(b) **Cost Dimension with Time-Factor 10**



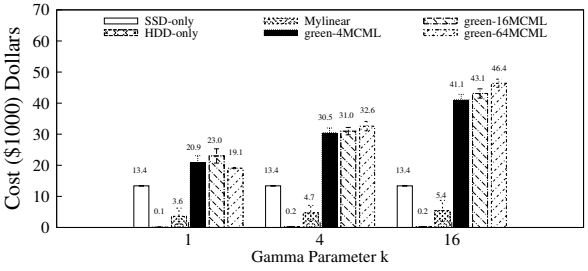
(c) **Cost Dimension with Time-Factor 100**



(d) **Cost Dimension with Time-Factor 1,000**



(e) **Cost Dimension with Time-Factor 10,000**



(f) **Cost Dimension with Time-Factor 100,000**

Figure 7.3: Fileserver workload results. We scale the time factor to show various results.

7.3.3 File-Server Workload

We present the results of file-server workload in Figure 7.3. As we can see from Figures 7.3(a), 7.3(b), 7.3(c), 7.3(d), 7.3(e), and 7.3(f), when the time factor is 1 and 10, the SSD-only drive based system tends to incur the highest cost, the HDD-only drive based system tends to incur the lowest cost, and tiering hybrid drive based systems tend to incur costs in the middle. However, as the time factor increases further, the incurred cost of the HDD-only drive increases the least, the incurred costs of GreenDM based systems increase the most, and the SSD-only drive and Mylinear drive based systems increase in the middle. Among the tiering hybrids, GreenDM causes more cost than Mylinear does. The larger the time factor is, the larger the difference is. When the time factor is 1, the difference is virtually zero, however, when the time factor is 1,000, the difference is maximally $510, and when the time factor is 100,000, the difference is maximally $41,000. The number of data migrations in GreenDM is the main reason. Note that, for fileserver workloads, when the time factor is 100,000, it translates to 1 year on average for all types of benchmarks (min and max are 0.2 and 2.4 years, respectively).

Different GreenDM MCML values under different Gamma values can incur different costs. The difference varies as the time factor vary. When the time factor is 1, the difference is virtually $0. When the time factor is 10, the difference is within $1. When the time factor is 100, the difference varies from $1 to $5. When the time factor is 1,000, the difference varies from $5 to $53 . When the time factor is 10,000, the difference varies from $100 to $500. When the time factor is 100,000, the difference varies from $500 to $5,300. The reason lies in the fact that different GreenDM MCML values under different Gamma values affect the trade-offs of performance, energy, and endurance in different ways as we discussed in Chapter 6.

For this workload, a large MCML value of 64 under smaller Gamma values tend to incur the least cost in the long run since: (1) a smaller Gamma value causes less number of data migrations; and (2) when Gamma value is small, a large MCML of 64 enables more I/Os to be served from OS page buffer. Therefore, system parameters have to be chosen carefully to better justify the gained performance.

7.3.4 Summary

We summarize the interesting observations across all workloads regarding the associated cost aspect of GreenDM as follows:

1. For the read-intensive Web-search workload, a larger PT value incurs less cost in the long run; for the write-intensive FIU online trace, a smaller ES value incurs less cost in the long run; for the file-server workload that has a different read/write ratio, a large MCML value when the Gamma value is small incurs the least cost in the long run.

2. The HDD-only drive based system has the least initial capital investment, and incurs the least cost in dollars in the long run. Note that the HDD-only drive based system has the lowest performance as well.

3. The SSD-only drive based system has the largest initial capital investment, and can incur low and high long-term costs for read-intensive and write-intensive workloads, respectively. Note that the SSD-only drive based system has the highest performance as well.

4. Tiering based hybrid systems have medium initial capital investment, and can incur costs in different degrees in the long run. Note that the tiering hybrids based system has a medium throughput as well.

5. Depending on the data management policy, the associated costs of different tiering hybrids can vary a lot in the long run: GreenDM triggers more cost than Mylinear does; however, GreenDM achieves better performance than Mylinear does.

6. Different GreenDM configurations lead to a certain degree of difference in cost, which increases as the time factor increases. The difference also largely depends on the workloads.

7.4 Related Work

Gartner reported that the total cost of ownership is 5–10$\times$ that of the initial purchase cost for storage systems [42]. However, the study is old (1999) and does not consider modern deployment using SSDs. More recently, Gartner performed additional long-term cost studies for desktops [43] and notebooks [44]; however, these studies are only for the computers (i.e., hardware and software purchase and update), not dedicated for storage systems, and they do not consider device replacement cost.

There are several related studies on the cost dimension of storage systems with SSD deployed. Some of them are using simulation [68,103], instead of performing realistic experiments. Some [49, 92] do not consider the SSD endurance cost in their total cost calculation. The cost dimension of storage systems with SSD deployed is discussed a lot in industry as well [38,123], however, detailed cost model that considers the total cost of ownership is not publicly available.

Our work is different in several aspects: (1) we collect realistic energy and power numbers from experiments; (2) we calculate the SSD endurance cost as well; (3) we scale the experiments to observe long-term effects; and (4) We developed and discussed a cost model containing the total cost of ownership.

7.5 Limitations

Modeling the cost dimension of storage systems is not an easy task. There is certainly limitations regarding our current cost model. Specifically, we are not considering the following cost aspects yet: (1) computer resource cost; (2) air conditioning cost; (3) labor power cost; and (4) varying service cost. We are also simplifying several conditions to make the cost model easy to understand: (1) the hardware setup in a real data center may be more complex than our setup in the experiments; (2) the service cost may not be fixed; and (3) the workloads in a real data center may be more complex than the workloads we used in the experiments. Therefore, our cost model approach is more like a best-effort approach. However, our work could potentially help build more accurate cost models to better justify the performance gained in storage systems.

7.6 Conclusion

We have presented our cost model to show several interesting results. The results further prove that the trade-offs among performance, energy, and device endurance play a role in the aspect of economics. Future storage system designs have to consider multiple optimizations: performance, energy, endurance, and cost.

Chapter 8

Caching Follow-Up

Tiering and caching based hybrid approaches share several design traits with each other (e.g., similar data management policies). But, they are not the same approaches. There are existing studies [24, 52] exploring the pros and cons of the tiering and caching based approaches. However, there is no current work that builds the two realistic systems with similar strategies, and empirically evaluates the two systems from the cost perspective under the same environment, when SSDs are deployed.

Our environment setup is not the best case for a caching system, but for a tiering system. However, we choose to delve into such a caching system, in comparison with the tiering under the same hardware and similar software setup, to provide more interesting observations. We observed that for some workloads, using the SSD as a cache had lower costs than when the SSD was used as primary hot-data storage; but other workloads completely reversed this trend.

We discuss the design and implementation of the caching system in Section 8.1. We evaluate the caching system in Section 8.2. We discuss related work on tiering and caching in Section 8.3. We explain the limitations of the caching system in Section 8.4. We conclude the caching follow-up work in Section 8.5.

8.1 Design and Implementation

Our caching system is largely based on the tiering system as we previously discussed in Section 6.1. Understanding how the tiering system works helps a lot in the discussion because the two are very similar to each other. For better illustration, we present the architecture of our caching system in Figure 8.1, and show the data management strategies in Figure 8.2. To distinguish the caching system from the tiering system, we explain the differences between the two systems below.

Capacity In the caching system, since the SSD is not counted toward the total capacity, the HDD capacity needs to be expanded to yield the same amount of total capacity as the tiering system has. When the SSD capacity is not largely different than the total capacity, a tiering system can have better purchase cost per GB than the caching system does.

Management Unit The caching system uses a cache entry table and the tiering system uses a mapping table. Unlike the tiering mapping table that maps from the whole virtual layer to the

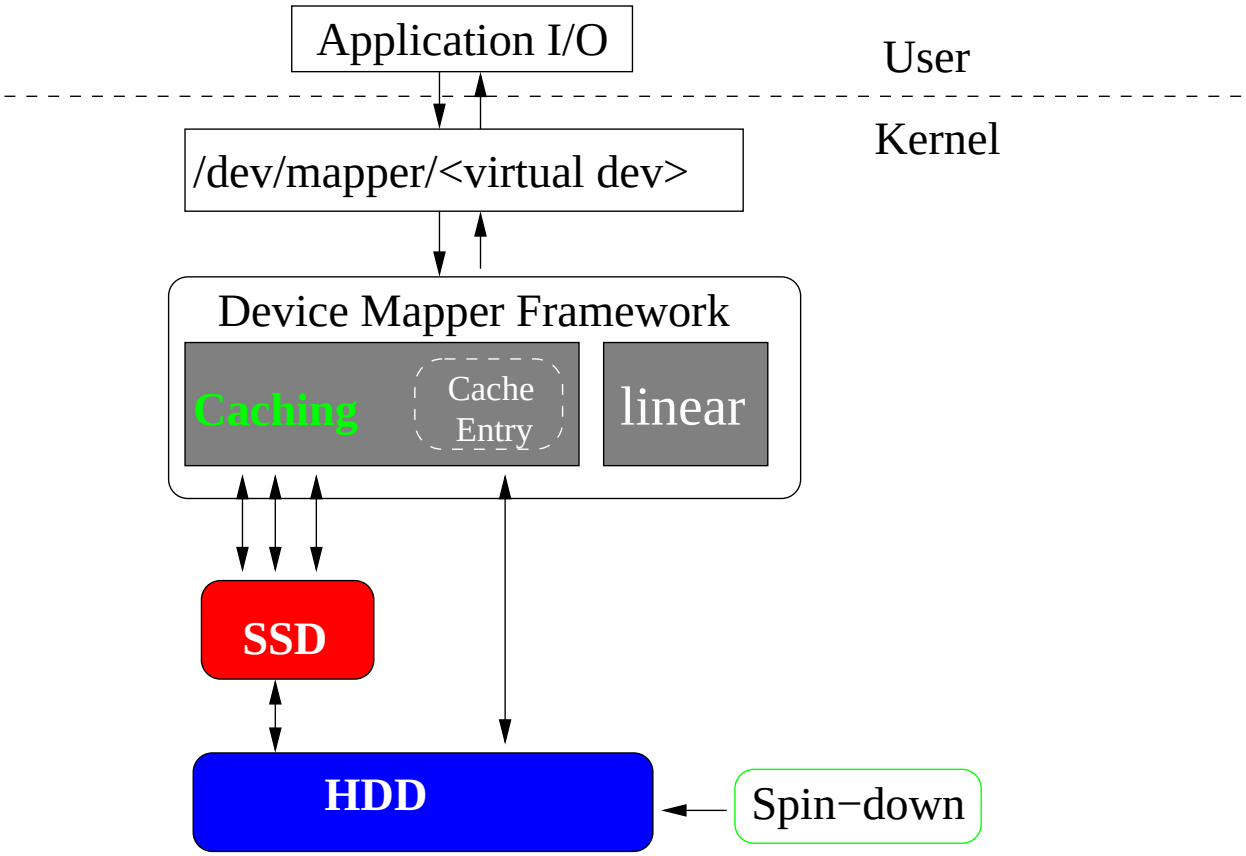


Figure 8.1: **Caching Architecture**. *The shaded rectangles are DM targets, usually implemented as loadable kernel modules.*

physical layer, the cache entry table maintains mapping information only from the cache device to the lower-level device. It contains not only the four fields in the mapping table of the tiering system (i.e., extent ID, state, usage counter, and time-stamp of the latest access), but also a dirty flag to indicate whether a cached extent is updated or not.

Data Movement The two systems use the same method to move data around. We name the hot data moving process *promotion* and *pre-fetch* in the tiering system and caching system, respectively. We name the cold data moving process in the two systems *demotion* and *eviction*, respectively. The caching system does not need to reserve extra extents in the HDD for eviction to succeed, as it is guaranteed to map an extent from the SSD to the HDD.

Read/Write Policy In a tiering system, since the SSD is used as primary storage, reads and writes access the data from the current location either on the SSD or HDD according to the mapping table. Cold data migrates to the HDD and hot data eventually migrates to the SSD using kernel threads. In a caching system, reads and writes access data from the SSD if the data is still there, else from the HDD. If it is an SSD write hit or if there is a write to any of the I/Os that are served from RAM, the system stores information of the pending write-back I/O in a queue, and an asynchronous write-back kernel thread wakes up to flush dirty writes from the SSD to the HDD. To control the rate

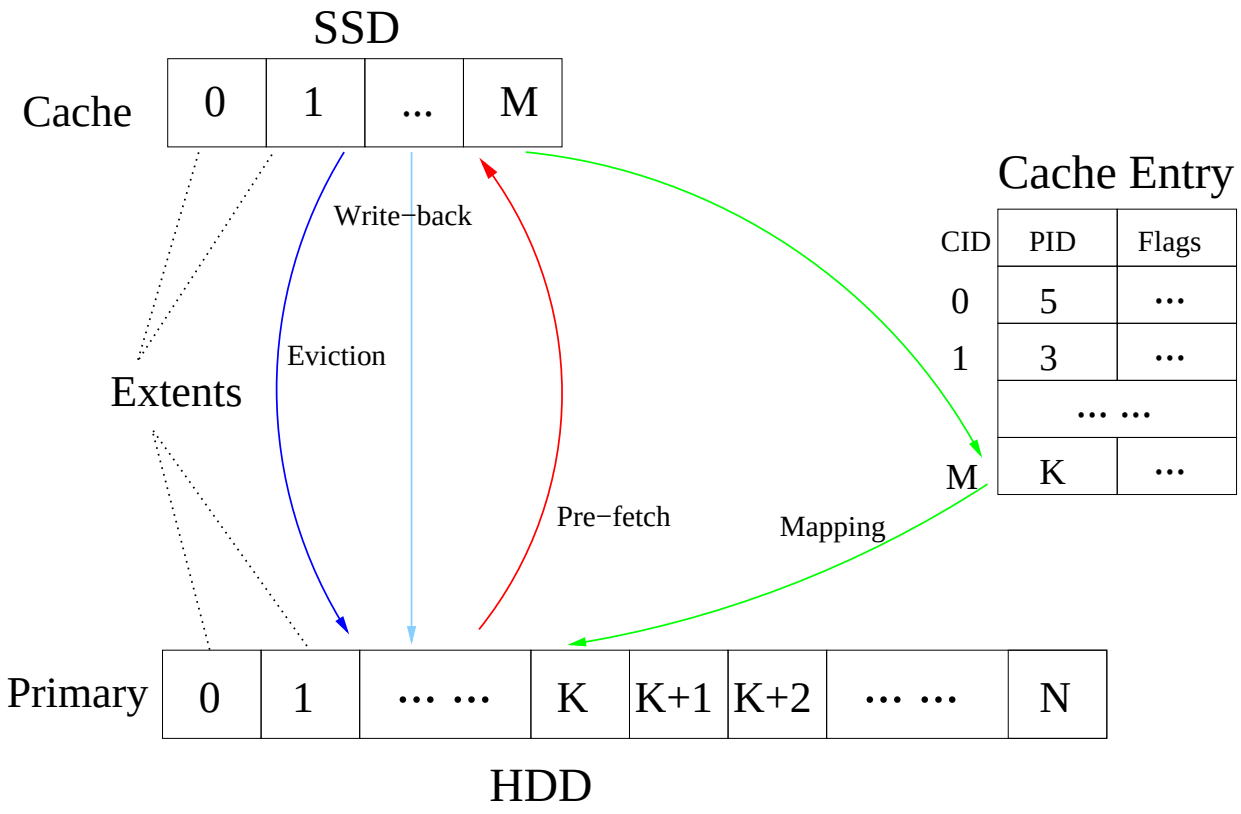


Figure 8.2: **Data Management in Caching**. *"CID" and "PID" represent the cache extent ID and the primary extent ID, respectively. The green arrows illustrate the mapping from the cache device to the primary device.*

of the write-back process, our caching system removes duplicated, queued write-back I/Os and caps the maximum queue size. The queue is not blocking the whole system because the queue is around $1/3$ full under pressure. The current policy can help illustrate the negative effects of the caching write-back policy compared with the tiering system that requires no write-back at all. The implementation detail of the caching's write policy is based on the eviction process for easy data management. I/O access can be slow down during write-back activity.

Other than the differences as we explained above, the two systems are virtually identical in terms of design and implementation.

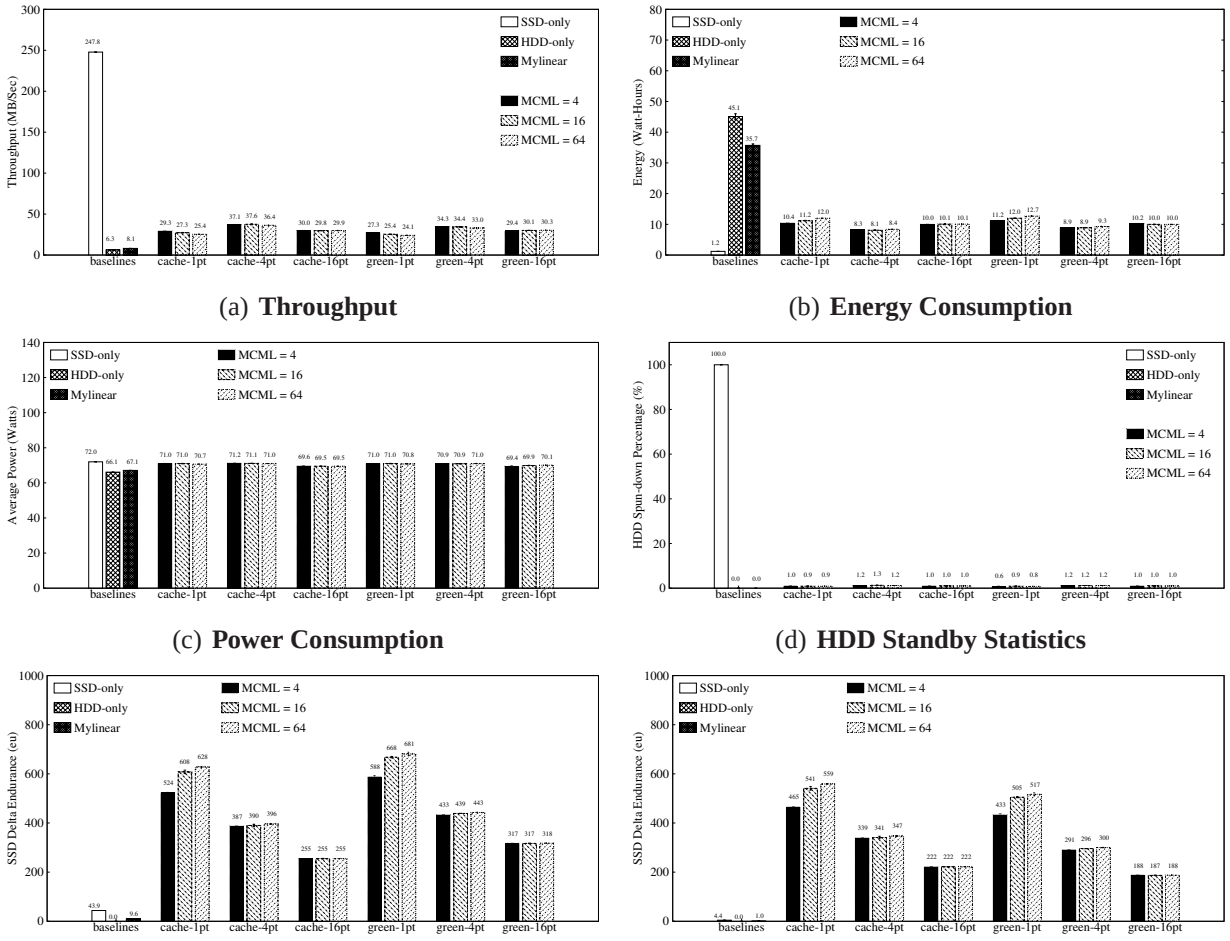
8.2 Evaluation

Our caching system is evaluated under the exactly same setup as we do for the tiering system, as discussed in Section 6.2. Therefore, we do not duplicate the setup description in this evaluation section for the caching system. The front tier device sizes (i.e., SSD size used in the experiments) are the same as well to produce comparable results.

We then discuss the results of the caching system under the same workloads: (1) Web-Search trace workload; (2) FIU's online trace workload; and (3) Filebench file-server workload. The cost results are based on the same cost model as we discussed in Chapter 7.1. For clarity, we use

abbreviations in the results below: (1) MCML means Maximum Concurrent Migration Limit; (2) PT means Pre-fetching Threshold (similar to Promotion Threshold); and (3) ES means Extent Size.

8.2.1 Web-Search Trace Workload



(a) **Throughput**

(b) **Energy Consumption**

(c) **Power Consumption**

(d) **HDD Standby Statistics**

(e) **SSD Delta Endurance with Ratio** $1/10$

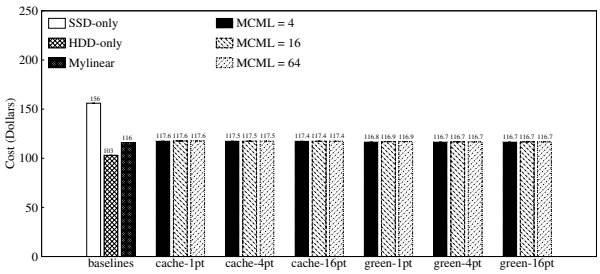
(f) **SSD Delta Endurance with Ratio** $1/100$

Figure 8.3: **Web-Search Trace Replay Results (part 1)**.

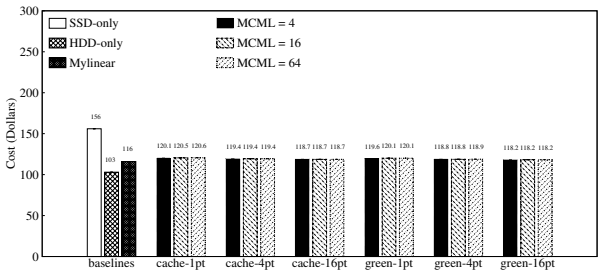
We present the Web-search trace workload results of the caching system in Figures 8.3 and 8.4. The parameter configurations for the caching system is the same as the tiering system has for the Web-search trace workload.

As shown in Figures 8.3(a), 8.3(b), 8.3(c), 8.3(e), and 8.3(f), We can observe similar trade-offs among performance, energy, power, and endurance as we discussed for our tiering system in Section 6.2.3. In Figures 8.4(a), 8.4(b), 8.4(c), 8.4(d), 8.4(e), and 8.4(f), we can observe similar trend for the total cost of the system as we discussed for our tiering system in Section 7.3.1.

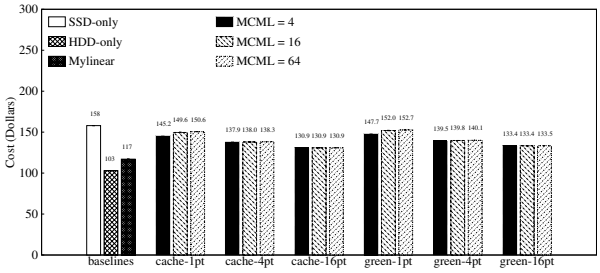
We then compare the caching system against the tiering system in more details. For this Web-search trace workload, the caching system achieves slightly higher throughput (i.e., 4–9%) than the tiering system does when the Pre-fetching Threshold (PT) is 4 and 16; and achieves very similar



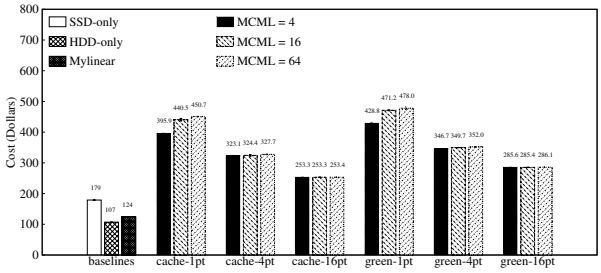
(a) **Cost Dimension with Time-Factor 1**



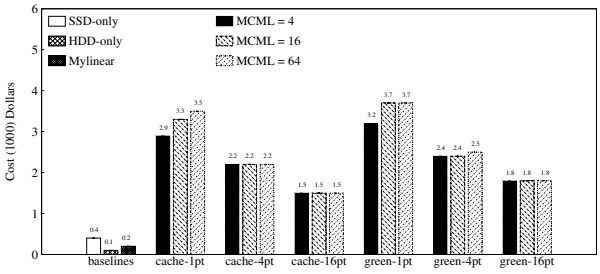
(b) **Cost Dimension with Time-Factor 10**



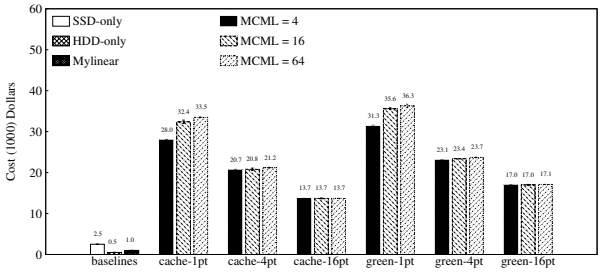
(c) **Cost Dimension with Time-Factor 100**



(d) **Cost Dimension with Time-Factor 1,000**



(e) **Cost Dimension with Time-Factor 10,000**



(f) **Cost Dimension with Time-Factor 100,000**

Figure 8.4: **Web-Search Trace Replay Results (part 2)**.

throughput than the tiering system does when PT is 64, as we can see in Figure 8.3(a). For the purpose of explanation, the Web-search trace workload has much more reads than writes, as shown in Table 6.3. That means the overhead of the write back is not going to be significant since there are only a few writes. Moreover, as the primary storage (i.e., SSD) in the tiering system contains either cold or hot data before hand, this can incur additional overhead to the overall throughput. However, the caching device in the caching system only contains hot data. That means the overall throughput of the caching system may be higher than that of the tiering system in some degree if the primary storage initially contains cold data in the tiering system.

The caching system also consumes very similar energy as the tiering system does when the PT is 1 and 16. When the PT is 4, the caching system consumes slightly less energy (i.e., 7–10%) than the tiering system does, as shown in Figure 8.3(b). The main reason is that the energy consumption is coupled with the throughput since the total amount of workload is the same.

The caching system consumes similar power as the tiering system does, as shown in Figure 8.3(c). The caching system also rarely spins down the HDD, as shown in Figure 8.3(d).

Besides, the caching system also incurs less SSD endurance reduction (i.e., 8–20%) than that of the tiering system when the ratio of read-to-write effect is $1/10$, as seen in Figure 8.3(e). The reason is due to the aggregated primary SSD I/Os in the tiering system. The caching system wears out the SSD faster (i.e., 8–19%) than the tiering system does when the ratio of read-to-write effect is $1/100$, as we can see in Figure 8.3(f). The reason is two-fold: (1) this Web-search is read-dominated; and (2) reads are not as effective as the writes to reduce the SSD endurance.

Moreover, the caching system causes less total dollar cost in the long run (i.e., 8–20%) than the tiering system does, as shown in Figure 8.4(f). Note that for Web-search workload, when the time factor is 100,000, it translates to 2.1 years on average (min and max are 0.2 and 7.7 years, respectively) for all types of benchmarks, a reasonable time frame for long-term storage. The reason for the above difference is that the caching system wears out the SSD slower when the ratio of read-to-write effect is $1/10$, and the current cost results are based on the assumption that the ratio of read-to-write effect is $1/10$.

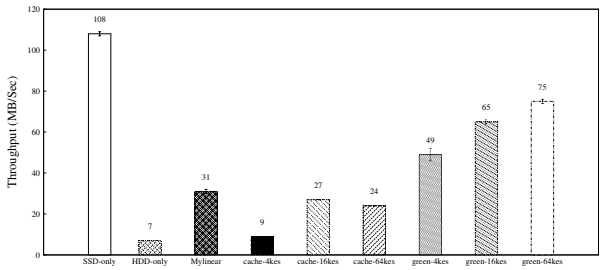
8.2.2 Online Trace Workload

We present the FIU online trace workload results of the caching system in Figures 8.5 and 8.6. The parameter configurations for the caching system is the same as the tiering system has for the online trace workload.

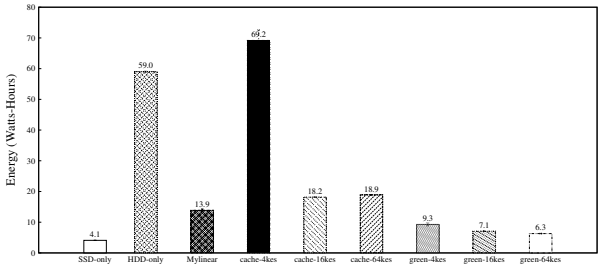
As shown in Figures 8.5(a), 8.5(b), 8.5(c), 8.5(e), and 8.5(f), We can observe similar trade-offs among performance, energy, power, and endurance as we discussed for our tiering system in Section 6.2.4. In Figures 8.6(a), 8.6(b), 8.6(c), 8.6(d), 8.6(e), and 8.6(f), we can observe similar trend for the total cost of the system as we discussed for our tiering system in Section 7.3.2.

We then compare the caching system against the tiering system in more details. For this FIU online trace workload, the caching system achieves less throughput (i.e., 58–82%) than the tiering system does when the ES varies, as we can see in Figure 8.5(a). For the purpose of explanation, the online trace workload has lots of writes, as shown in Table 6.3. That means the overhead of the write back can be a bottleneck when it comes to throughput.

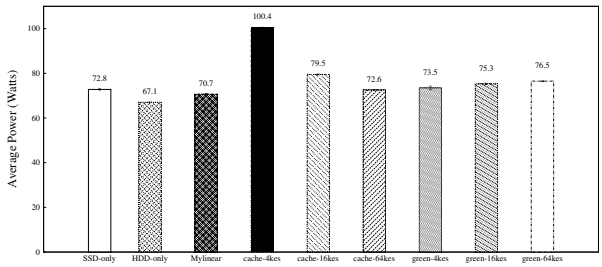
The caching system also consumes more energy (i.e., 2–6$\times$) than the tiering system does, as shown in Figure 8.5(b). The reason is that the total energy consumption is coupled with the throughput and the total amount of workload is the same.



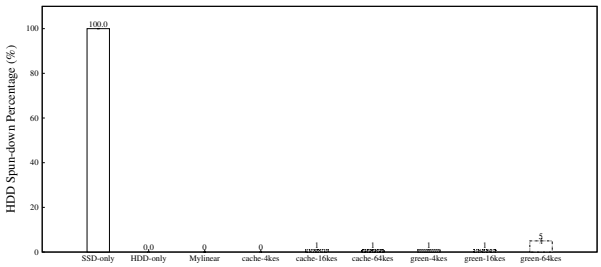
(a) **Throughput**



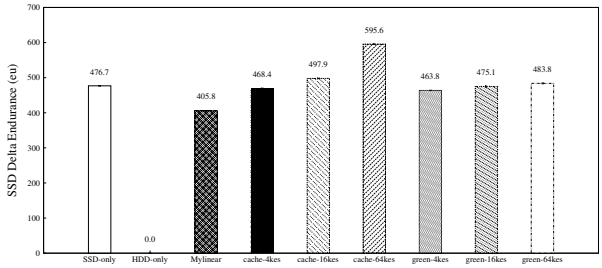
(b) **Energy Consumption**



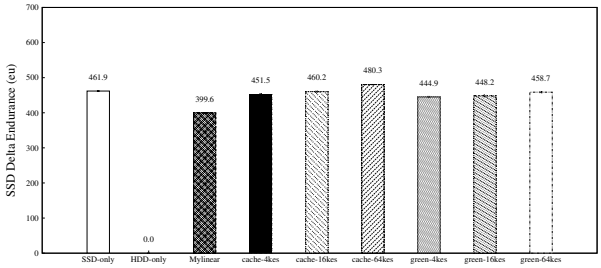
(c) **Power Consumption**



(d) **HDD Standby Statistics**

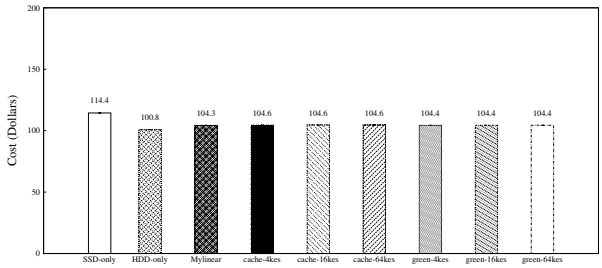


(e) **SSD Delta Endurance with Ratio** $1/10$

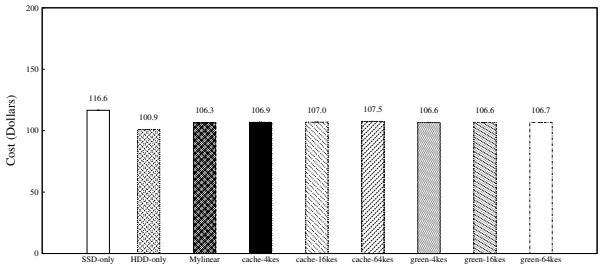


(f) **SSD Delta Endurance with Ratio** $1/100$

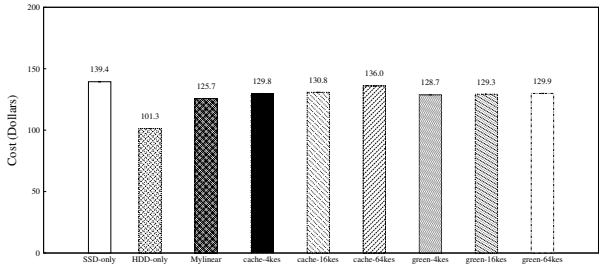
Figure 8.5: **Online Trace Replay Results (part 1)**.



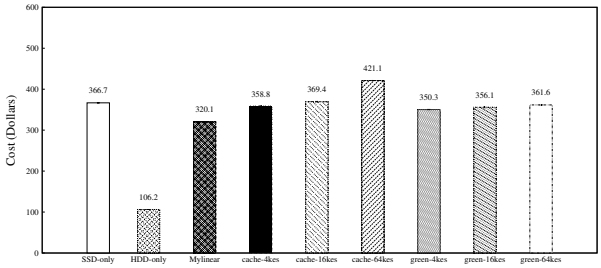
(a) **Cost Dimension with Time-Factor 1**



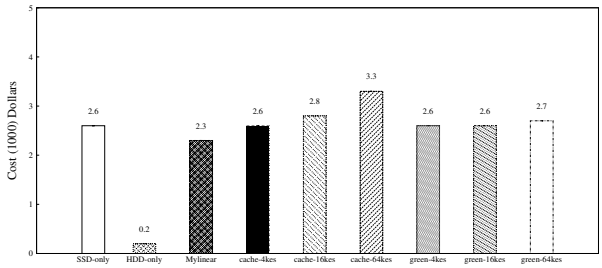
(b) **Cost Dimension with Time-Factor 10**



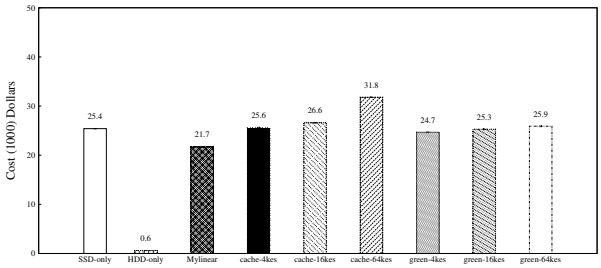
(c) **Cost Dimension with Time-Factor 100**



(d) **Cost Dimension with Time-Factor 1,000**



(e) **Cost Dimension with Time-Factor 10,000**



(f) **Cost Dimension with Time-Factor 100,000**

Figure 8.6: **Online Trace Replay Results (part 2)**.

The caching system consumes similar power as the tiering system does, as seen in Figure 8.5(c). The only big difference is that for the caching system, when the ES is 4K, the caching system consumes much higher power than other conditions. The reason is that when the ES is 4K, there are much more write-back I/Os that cause the system (i.e., CPU, RAM, and I/O system) to be even more active. The caching system also rarely spins down the HDD, as shown in Figure 8.3(d).

Besides, the caching system also incurs more SSD endurance reduction (i.e., 1–23% and 2–5%) than the tiering system does when the ratio of read-to-write effect is $1/10$ and $1/100$, respectively, as seen in Figure 8.5(e) and Figure 8.5(f), respectively. The main reason is that the caching system has more write-back I/Os than the aggregated primary SSD I/Os in the tiering system. Therefore, the caching system can wear out the SSD faster than the tiering system does.

Moreover, the caching system causes more total dollar cost in the long run (i.e., 5–23%) than the tiering system does, as shown in Figure 8.6(f). Note that for FIU online workload, when the time factor is 100,000, it translates to 3.3 years on average for all types of benchmarks (min and max are 0.7 and 9.8 years, respectively). The reason for the above difference is that the caching system wears out the SSD faster as we explained above, and the SSD endurance reduction counts more toward the total cost of ownership.

8.2.3 File-server Workload

We present the file-server workload results of the caching system in Figures 8.7 and 8.8. The parameter configurations for the caching system is the same as the tiering system has for the file-server workload.

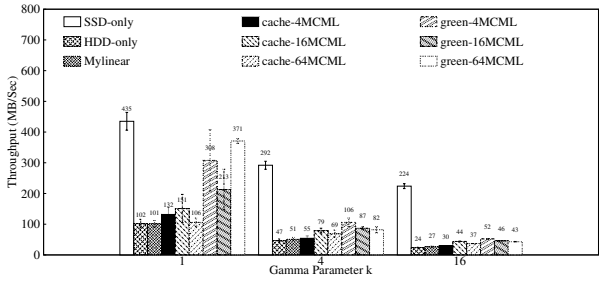
As shown in Figures 8.7(a), 8.7(b), 8.7(c), 8.7(e), and 8.7(f), We can observe similar trade-offs among performance, energy, power, and endurance as we discussed for our tiering system in Section 6.2.5. In Figures 8.8(a), 8.8(b), 8.8(c), 8.8(d), 8.8(e), and 8.8(f), we can observe similar trend for the total cost of the system as we discussed for our tiering system in Section 7.3.3.

We then compare the caching system against the tiering system in more details. For this Filebench file-server workload, the caching system achieves less throughput (i.e., 14–71%) than the tiering system does when the system parameters (i.e., MCML, PT, and Gamma) vary, as we can see in Figure 8.7(a). For the purpose of explanation, the file-server workload has lots of I/Os: both reads and writes. There are more reads than writes, but the difference is not that significant, according to Filebench. That means the overhead of the write back is going to play some role in making the throughput lower.

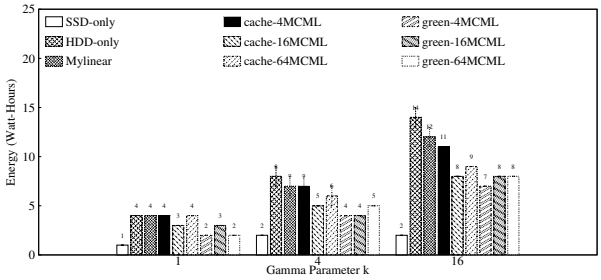
The caching system also consumes more energy (i.e., up to 57%) than the tiering system does, as shown in Figure 8.7(b). The reason is similar with what we have explained in Section 8.2.1.

The caching system consumes similar power as the tiering system does, as shown in Figure 8.7(c). The caching system spins down the HDD even less than the tiering system does, as shown in Figure 8.7(d) and since the caching system uses the HDD more aggressively than the tiering system.

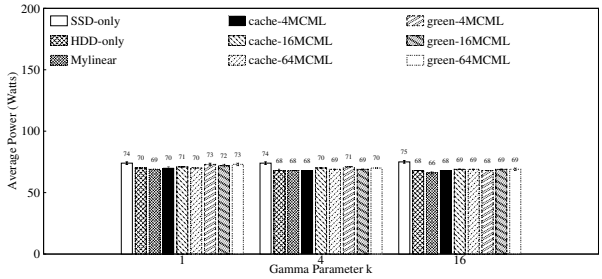
Moreover, the caching system wears out the SSD slower (i.e., 27–60% and 28–60%) than the tiering system does when the ratio of read-to-write is $1/10$ and $1/100$, respectively, as seen in Figure 8.7(e) and Figure 8.7(f), respectively. The main reason is that the tiering system's aggregated primary SSD I/Os become a bigger factor toward the SSD endurance reduction. Therefore, the caching system wears out the SSD slower than the tiering system does.



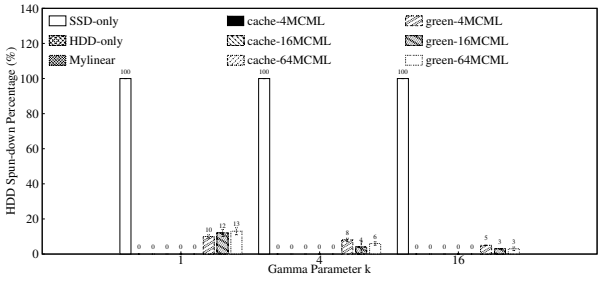
(a) **Throughput**



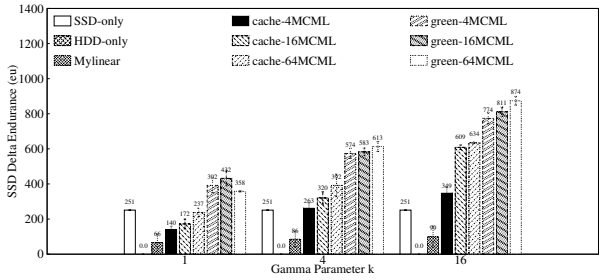
(b) **Energy Consumption**



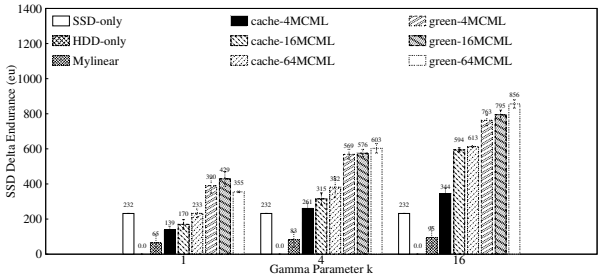
(c) **Power Consumption**



(d) **HDD Standby Statistics**

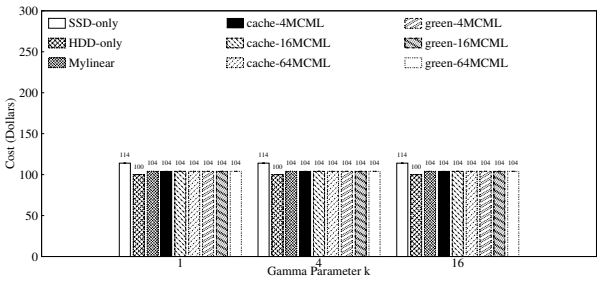


(e) **SSD Delta Endurance with Ratio** $1/10$

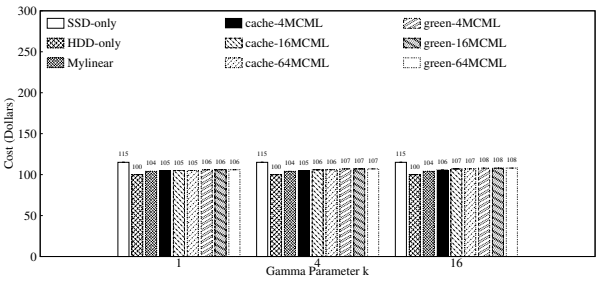


(f) **SSD Delta Endurance with Ratio** $1/100$

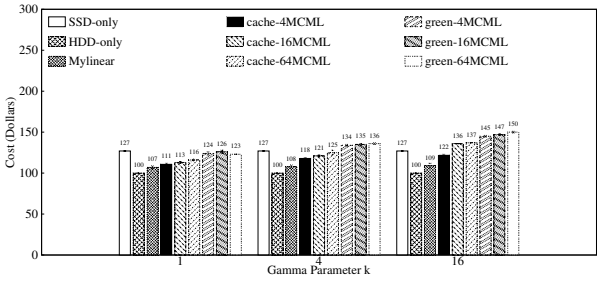
Figure 8.7: **Fileserver Workload Results (part 1)**.



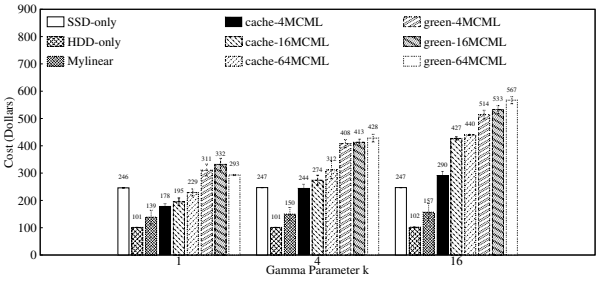
(a) **Cost Dimension with Time-Factor 1**



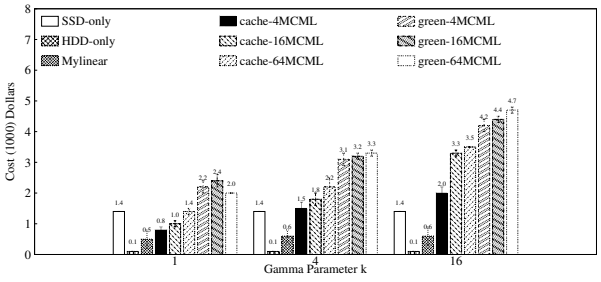
(b) **Cost Dimension with Time-Factor 10**



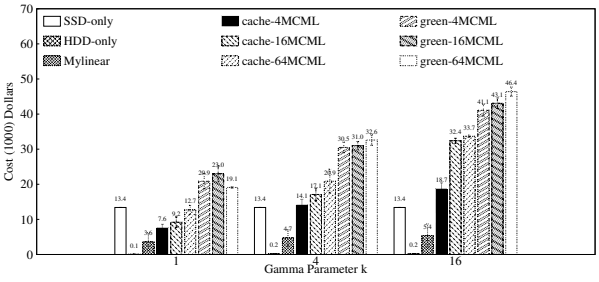
(c) **Cost Dimension with Time-Factor 100**



(d) **Cost Dimension with Time-Factor 1,000**



(e) **Cost Dimension with Time-Factor 10,000**



(f) **Cost Dimension with Time-Factor 100,000**

Figure 8.8: **Fileserver Workload Results (part 2)**.

Moreover, the caching system causes less total dollar cost in the long run (i.e., 27–55%) than the tiering system does, as shown in Figure 8.8(f). Note that for the fileserver workloads, when the time factor is 100,000, it translates to 1 year on average for all types of benchmarks (min and max are 0.2 and 2.4 years, respectively). The main reason for the above difference is that the caching system incurs less SSD endurance reduction as we explained above, and the SSD endurance reduction counts more toward the total cost of ownership.

8.2.4 Summary

We summarize the caching follow-up results in comparison with the tiering system below.

1. For the read-intensive Web-search workload, caching achieves similar throughput, energy and power consumption, and short-term dollar cost with the tiering system since the two systems are similar. Moreover, caching wears out the SSD slower and has lower long-term dollar cost than tiering due to the aggregated primary SSD I/Os in the tiering system.

2. For the write-intensive Online workload, caching achieves less throughput than tiering does due to the negative effect of the write-back policy in the caching system. Moreover, caching causes higher energy and power consumption, wears out the SSD faster, and causes higher long-term cost than tiering does due to the same write-back negative effect in the caching system. Besides, caching causes similar short-term cost than tiering does since the hybrid drive setup is relatively small.

3. For the read- and write-intensive file-server workload, caching achieves less throughput than tiering does due to the write-back negative effect in the caching system. Moreover, caching also wears out the SSD slower and incurs less long-term cost because the tiering system has more aggregated primary SSD I/Os. Moreover, caching causes higher energy consumption than tiering, and similar power consumption and short-term cost with tiering.

4. Caching and tiering are very similar except that: (1) caching only maintains mapping information from the cache device to the lower-level device while tiering has to maintain mapping from the whole virtual device to the actual physical devices; (2) caching has to further support a write policy in case of a write hit in the cache device while tiering does not need to; and (3) tiering can achieve better initial purchase cost over capacity than caching does since the hot device is used as the primary storage.

8.3 Related Work

There are several related work on the comparison between caching and tiering systems. MAID [24] briefly discusses the pros and cons of caching and migration based policies for massive storage systems. With the advent of Phase Change Memories (PCMs), there is one work [52] that evaluates PCMs for enterprise storage systems by case studies of caching and tiering approaches. However, there is no direct comparison study performed for the caching and tiering approaches from the perspective of total cost of ownership.

Our work is different from the above work in several aspects: (1) we build two realistic systems (i.e., tiering and caching) with similar strategies and environment to evaluate the pros and cons of

the caching and tiering based storage systems; and (2) we look into the total cost of ownership for the two systems with SSD deployed to provide even more interesting observations.

8.4 Limitations

Caching system and tiering system share several design traits. Our caching system is largely based on the tiering system. Other than the limitations we discussed in Section 6.4, there is also another limitation. Our experiment setup is a better environment for our tiering system, not for a caching system. Caching system is normally deployed in large storage systems where the caching tier is comparably small compared with the lower-level storage (e.g., 1 PB). We stick to the current environment because our comparison study will only make sense when the hardware and software setup is the same between the two systems. It would be interesting and challenging to further discuss the caching system in a large storage system.

8.5 Conclusion

Caching system and tiering system are two types of hybrid storage systems. They share several design traits and are normally deployed in different contexts: caching system is normally explored in very large storage systems with relatively small caching size, and tiering system is normally utilized in relatively small storage system, where the capacity of the high-level tier storage is comparable with the lower-level storage.

To provide comparable results for the two hybrid systems, we designed, implemented, and evaluated the two systems in similar environment. However, we still observed interesting results. The results, collected under our current environment where tiering system can achieve better purchase cost over capacity, provide several interesting observations. First of all, tiering system generally achieves better performance than caching system since the write policy of the caching system incurs more overhead to the system and the tiering system tends to aggregate hot data more efficiently than the caching system. Secondly, the caching system can wear out the SSD faster or slower than the tiering system depending on two factors: (1) the SSD in caching system is not used as primary storage so that there is no primary SSD I/Os; and (2) there are a certain amount of write back I/Os in the caching system. Last but not least, caching system incurs more or less total cost (i.e., purchase cost plus total cost of ownership) than tiering systems in the long run depending on the SSD endurance reduction each system causes.

Chapter 9

Capacity Ratio Follow-Up

The capacity ratio of the SSD as a fraction over the total capacity matters for both the tiering and caching systems in terms of throughput, energy and power, device endurance reduction, and dollar cost. Previously, in Chapters 6, 7, and 8, we used $1/4$ as the capacity ratio of SSD over total. Here, we report results from trying $1/8$ as the SSD's capacity ratio over total capacity. We reran all experiments and report our results and analysis here.

In this Chapter, we present our evaluation results in Section 9.1, and summarize the capacity ratio discussion in Section 9.2.

9.1 Evaluation

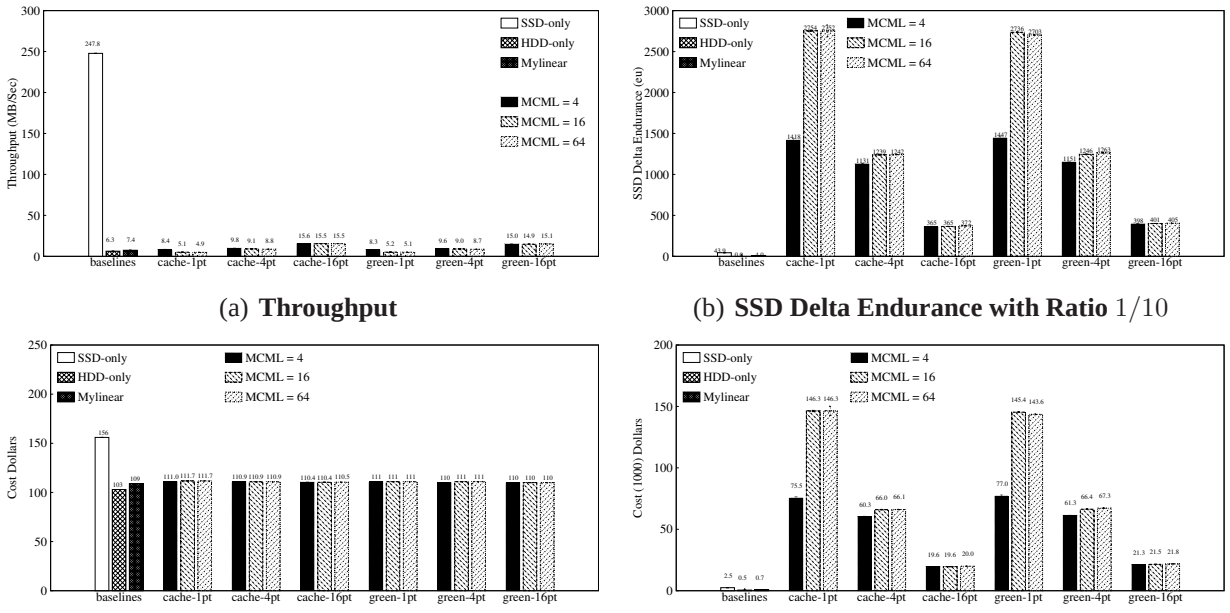
We reran all the experiments under the same hardware and software configurations, other than with a new $1/8$ capacity ratio of the SSD over total capacity. We present and discuss those results in this Section. To avoid duplicated description and discussion, we focus more on representative results (i.e., throughput, SSD endurance reduction, and cost).

9.1.1 Web-Search Trace Workload

We present the results of Web-search workload in Figure 9.1.

In terms of throughput for the Web-search workload, as we can see from Figures 8.3(a) and 9.1(a), when the SSD capacity ratio varies from $1/4$ to $1/8$, the throughputs for both caching and tiering goes down, ranging from 48% to 81%. The main reason for the throughput degradation is due to the increased number of data movements between the SSD and the HDD for both the two systems and a reduced SSD hit ratio when the available SSD capacity is reduced in half. We can also see that when the SSD capacity decreases, the throughput of Mylinear increases by 19% because of fewer SSD hits.

In terms of SSD endurance reduction, as we can see from Figures 8.3(e) and 9.1(b), when the SSD capacity ratio varies from $1/4$ to $1/8$, it wears out the SSD faster for both caching and tiering systems, from 26% to $3\times$. The reason is that the smaller SSD capacity causes more data movements between the SSD and the HDD for both tiering and caching systems. The smaller the PT value is, the more the data movements are incurred because PT tends to be the dominating factor



(a) **Throughput**

(b) **SSD Delta Endurance with Ratio** $1/10$

(c) **Cost Dimension with Time-Factor 1**

(d) **Cost Dimension with Time-Factor 100,000**

Figure 9.1: **Web-Search Trace Replay Results**. We configured ES to 1MB in GreenDM, to help with sequential prefetching. HDD spin-down was enabled for all configurations. The new capacity ratio of SSD over total is $1/8$.

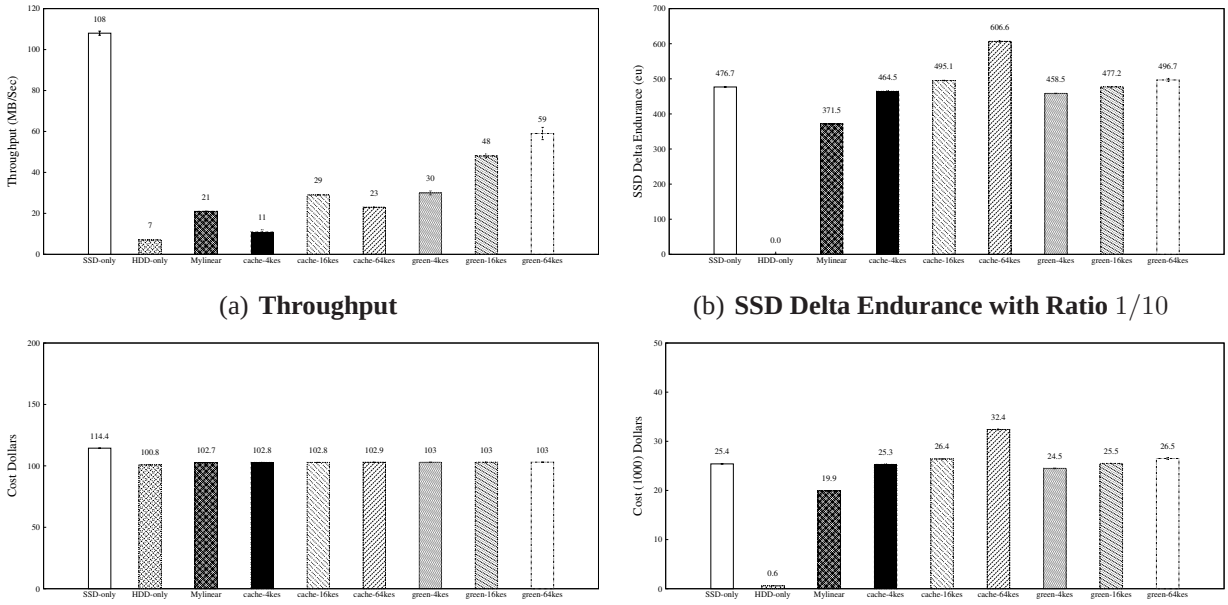
of SSD's endurance reduction. We can also see that when the SSD capacity decreases, it wears out the SSD 58% slower for Mylinear because of reduced SSD accesses.

In terms of cost, as we can see from Figures 8.4(a) and 9.1(c), and Figures 8.4(f) and 9.1(d), when the SSD capacity ratio varies from $1/4$ to $1/8$, it incurs similar short-term cost for both the caching and tiering systems, but incurs more long-term cost for both systems, from 27% to $3\times$. The reason for the similar short-term cost is due to the fact that when the capacity ratio varies, it only causes little variation in the capacity cost, energy and power costs, and the SSD replacement cost, when the time-factor is 1. The reason for the different long-term cost is because of the additional number of data movements between the SSD and the HDD. We can also see that when the SSD capacity ratio decreases, it incurs 30% less long-term cost for Mylinear due to reduced SSD accesses.

9.1.2 Online Trace Workload

We present the results of Online workload in Figure 9.2.

For throughput, as we can see from Figures 8.5(a) and 9.2(a), when the SSD capacity ratio varies from $1/4$ to $1/8$, the throughput for the tiering system goes down, from 21% to 39%; throughput for the caching system increases from 7% to 18% when the ES is 4K and 16K, respectively, and decreases by 4% when the ES is 64K. The reason for the tiering system throughput degradation is due to the increased data movements between the SSD and the HDD when the SSD capacity is smaller. The reason for the caching system throughput increase when the ES is small is because: (1) the caching system is bottlenecked by the write-back I/Os since this is a write-intensive



(a) **Throughput**

(b) **SSD Delta Endurance with Ratio** $1/10$

(c) **Cost Dimension with Time-Factor 1**

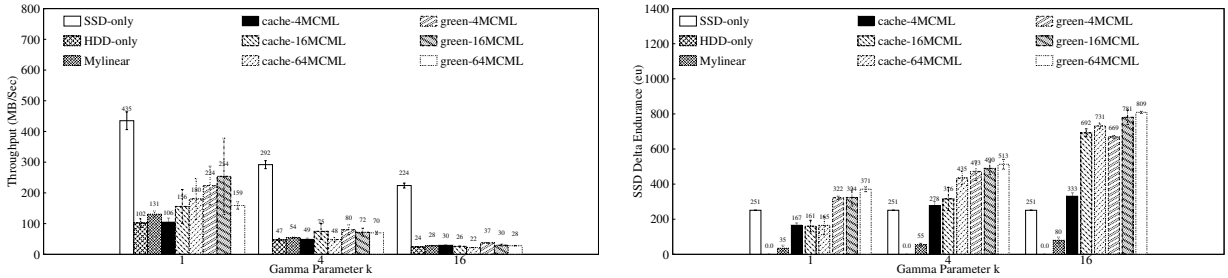
(d) **Cost Dimension with Time-Factor 100,000**

Figure 9.2: **Online Trace Replay Results**. As example, we set MCML to 16 and PT to 1. Disk spin-down was enabled for all configurations. The new capacity ratio of SSD over total is $1/8$.

workload; and (2) the smaller SSD capacity reduces the number of write-back I/Os. The reason for the small caching system throughput degradation is due to the fact that when the ES is larger, data movements between the SSD and the HDD causes more overhead. We can also see that when the SSD capacity decreases, the throughput of Mylinear increases by 32% due to the aforementioned reason.

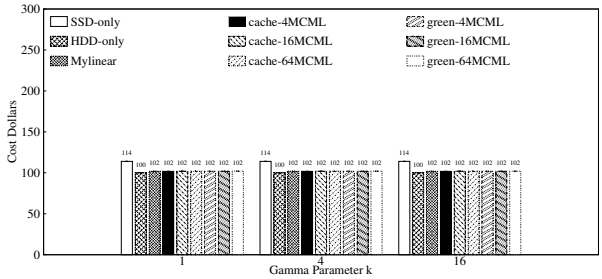
For SSD endurance reduction, as we can see from Figures 8.5(e) and 9.2(b), when the SSD capacity ratio varies from $1/4$ to $1/8$, the SSD endurance reduction for both tiering and caching stays roughly the same. The reason is actually due to the net effect of the following factors: (1) smaller SSD capacity leads to more data movements, which wears out the SSD faster; and (2) smaller SSD capacity leads to fewer SSD hits, which wears out the SSD slower. We can also see that when the SSD capacity reduces, it wears out the SSD 8% slower for Mylinear due to reduced SSD accesses.

For cost, as we can see from Figures 8.6(a) and 9.2(c), and Figures 8.6(f) and 9.2(d), when the SSD capacity ratio varies from $1/4$ to $1/8$, it incurs similar short-term and long-term costs for both tiering and caching systems. The reason for the similar short-term cost when the SSD capacity ratio reduces is due to the aforementioned reason. The reason for the similar long-term cost when the SSD capacity ratio reduces is due to the fact the SSD endurance reduction stays similar when the SSD capacity ratio varies. We can also see that when the SSD capacity decreases, it causes 8% less long-term cost for Mylinear because of reduced SSD accesses.

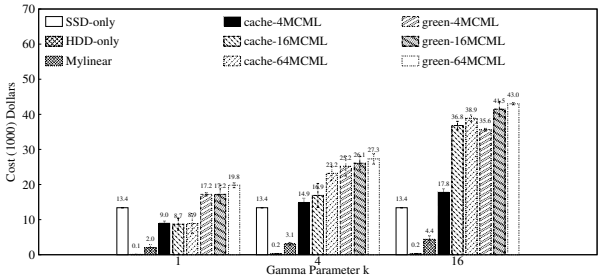


(a) **Throughput**

(b) **SSD Delta Endurance with Ratio** $1/10$



(c) **Cost Dimension with Time-Factor 1**



(d) **Cost Dimension with Time-Factor 100,000**

Figure 9.3: **Fileserver Workload Results**. We configured the ES to be 128KB in GreenDM. It is equal to the average I/O size to avoid the migration waste and I/O split overhead. The PT value was fixed at 1, as example. Disk spin-down was enabled for all. The new capacity ratio of SSD over total is $1/8$.

9.1.3 File-server Workload

We present the results of File-server workload in Figure 9.3. Note that the OS buffer cache is enabled for this workload.

In terms of throughput, as shown in Figures 8.7(a) and 9.3(a), when the SSD capacity ratio varies from $1/4$ to $1/8$, the throughputs for both tiering and caching systems generally reduces, from 35% to 57%. The main reason is due to the reduced SSD hits and increased data movements between the SSD and the HDD when the available SSD capacity becomes smaller. We can also see that when the SSD capacity decreases, the throughput of Mylinear increases from 4% to 30% instead of decreasing. We believe that the OS buffer cache is used to different degrees here, which causes this difference.

In terms of SSD endurance reduction, as shown in Figures 8.7(e) and 9.3(b), when the SSD capacity ratio varies from $1/4$ to $1/8$, it generally wears out the SSD slower for both tiering and caching, from 7% to 25%. The main reason is because there are fewer SSD hits and it is generally a bigger factor than the increased data movements. We can also see that for Mylinear, when the SSD capacity reduces, it wears out the SSD slower, from 19% to 47%, because of fewer SSD hits since the SSD capacity becomes smaller.

In terms of cost, as shown in Figures 8.8(a) and 9.3(c), and Figures 8.8(f) and 9.3(d), when the SSD capacity ratio varies from $1/4$ to $1/8$, it incurs similar short-term cost for both tiering and caching systems due to the aforementioned reason. It also generally incurs less long-term cost for both tiering and caching systems, from 7% to 25%. The reason is due to the fact that, when the SSD

capacity reduces, it wears out the SSD slower. There are cases where the long-term cost increases instead, and that is because the smaller SSD capacity causes a higher energy consumption which eventually increases the long-term cost. We can also see that when the SSD capacity is smaller, it incurs less long-term cost for Mylinear, from 19% to 44%, due to the net effect of reduced SSD access cost and increased energy cost.

9.2 Summary

We summarize the new SSD capacity ratio results in comparison with the previous old SSD capacity ratio below.

1. For the read-intensive Web-search trace workload, the smaller SSD capacity leads to decreased throughputs, wears out the SSD faster, and leads to increased long-term costs for both tiering and caching systems. This is due to the decreased SSD usage to a small degree, but primarily due to a larger increase in data movements. It also leads to decreased throughput, wears out the SSD slower, and incurs lower long-term cost for Mylinear due to decreased SSD hit.

2. For the write-intensive Online trace workload, the smaller SSD capacity causes different throughput effects in tiering and caching systems. For tiering, the throughput decreases—while for caching, the throughput generally increases. This is because for this workload, the caching system is bottlenecked by the write-back I/Os while the tiering system is bottlenecked by the SSD hit rate and the data movement. The smaller SSD capacity also leads to similar SSD wear out rates and long-term cost for both tiering and caching systems. This is due to the net effect of the reduced SSD hit and the increased data movement. It also leads to decreased throughput, wears out the SSD slower, and incurs lower long-term cost for Mylinear due to reduced SSD hit.

3. For the read- and write- intensive File-server workload, the smaller SSD capacity generally leads to decreased throughputs, reduced SSD wear out, and lower long-term costs for both tiering and caching systems. This is due to the decreased SSD hit to a large degree and increased data movements to a smaller degree. It also leads to increased throughput for Mylinear due to the OS buffer cache effects, wears out the SSD slower, and incurs lower long-term cost for Mylinear due to reduced SSD accesses.

4. For all workloads, the smaller SSD capacity leads to similar short-term costs for all hybrid systems. This is due to the fact that a smaller SSD capacity causes little variation for capacity cost, energy and power costs, and SSD replacement cost.

Chapter 10

Future

There are also several interesting longer-term related research topics that are interesting to explore in the future. this dissertation. Some of the promising post-defense future work are as follows:

1. Automate the choosing of controllable parameters.
2. Research and develop a three-tier and then N-tier storage system.
3. Support security as a fourth dimension.
4. Support New Storage Devices.
5. Provide Control Support at the CPU Level.

We further discuss the future work in more details below.

10.1 Automation of the Control Knobs

This section discusses the most interesting and challenging topic on automating the selection and setting of the controllable knobs. Tackling the mentioned issue here can help achieve user perceived importance among performance, energy efficiency, and endurance of the tiering hybrid drive storage system under various workloads.

Our tiered hybrid drive storage system provides several controllable knobs that can help certain workloads. The work will be made even more useful by coming up with a method to automate the choosing of the control knobs so that users do not need to specify the control parameters on their own. For example, we may provide options for the users to decide which dimensions, among performance, cost, and endurance, are more important—including weights applied to each dimension—so that internal system parameters can be selected automatically to meet the requirements.

We will try to explore such automation strategy in the future to make the work more useful. More specifically, an ad-hoc algorithm-based approach may be explored to guide the design of automation. For example, depending on the trade-offs we observe, we may develop a simple user-level "governor" that can potentially use one of several techniques to decide how many cores to keep on/off and what the GreenDM parameters should be: Hill climbing [57], Markov chains [87],

Machine learning [86], Fast Fourier transform [34], Wavelet [137], and Control theory [155]). This is going to be a challenging research topic to explore as well.

10.2 Three-Tier and N-Tier Storage Systems

This section discusses another interesting post-defense topic on extending the previous two-tier hybrid drive storage system to include the network tier as well. Based on the three-tier system, one may further investigate and develop an N-tier storage system. The research work on this topic can potentially have more interesting results and provide valuable observations to future hybrid storage system for years to come.

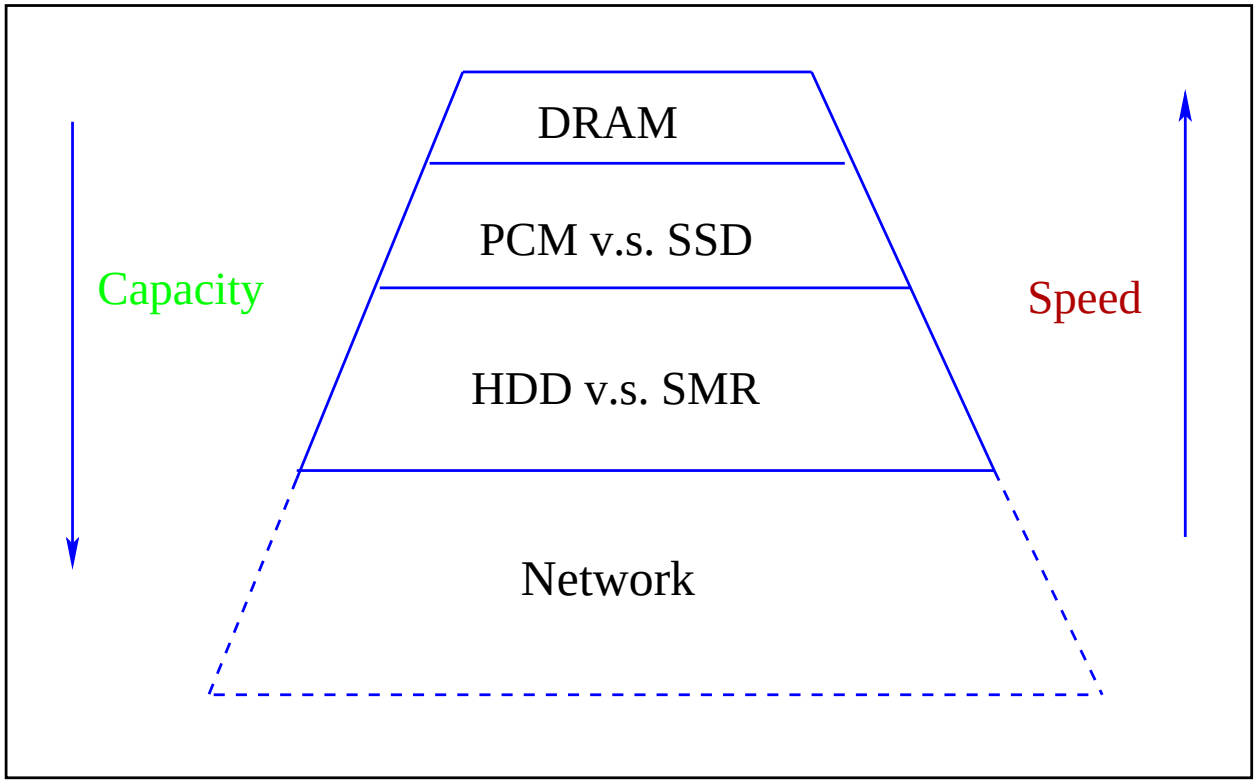


Figure 10.1: Storage pyramid

Our current system design, implementation, benchmarking, and analysis are based on a two-tier hybrid drive storage system. The interesting observations and results can be made even more useful by extending the two-tier hybrid drive storage system to a three-tier storage system, as shown in Figure 10.1. At the network tier, the space can be more plentiful (e.g., NAS and cloud storage), but network bandwidth becomes the bottleneck. It is also costly to flush data from a local to a remote site. Therefore, there are more important trade-offs to explore in terms of performance, cost, and endurance in the whole systems. Moreover, one can potentially further expand the two-tier storage system to be N-Tier ($N > 3$) to research more topics. Note that once the storage system has more tiers, it will become more complex to implement and analyze.

Moreover, each tier is flexible. It can be implemented in different layers and through different interfaces. For example, each tier is applicable either with block interface [129] or file inter-

face [64]. There are options both in kernel space [129] and in user space [67]. They have different trade-offs in deployment effort and versatility, however.

We will sure have much more policies in each tier to explore in the future, and there are several interesting questions to explore as well:

1. Should each tier be based on caching or tiering or even a mix of the two, and what are the trade-offs?

2. Which layer should each tier be implemented in? By which interface? What are the trade-offs?

3. How many controllable parameters would the three-tier or N-tier storage system have and how complex would the system become?

10.3 Security as a Fourth Additional Dimension

This section discusses considering security as a forth dimension of a hybrid storage system other than just considering the performance, energy efficiency, and endurance. This research topic can potentially open the door of a vast amount of future research topics.

Security is another important aspect of the storage system design. There are certainly trade-offs among the performance, energy, endurance, and security of a hybrid storage system. For example, to guarantee security, a hybrid storage system may add extra code to check the user's authentication and authorization, and to ensure data integrity and privacy. This can add an extra amount of overhead to the performance of the system, and can generate trade-off effects to the energy efficiency and endurance dimension of the whole system as well.

Exploring security as a fourth dimension would be an advanced and interesting topic to study in the future. Once security is in the list, users would have even more choices to explore. In particular, the users may consider which dimension among the four, is more important (and how much) under their particular deployment scenarios. For example, for performance-oriented application, the user can prioritize the control switch for performance, and for security-oriented applications, the user can prioritize the control switch for security.

An even more interesting topic would be having upper and lower caps on the security levels to control in a finer granularity. For example, a person may not be willing to reduce the security beyond a certain point, therefore, the system should be remain within its the caps.

10.4 Support New Storage Devices

This section details interesting future work on exploring new storage devices toward building cost-effective I/O services of storage systems. This topic can help discover more trade-offs of the new and yet less understood storage devices in terms of performance, energy efficiency, and endurance.

Hybrid system designs are becoming more popular in today's storage systems to provide cost-aware I/O services. Modern storage devices are emerging and evolving rapidly. They differ in speed, capacity, cost, endurance, power consumption, and more.

New storage devices provide more flexibilities to the layout of each tier in a hybrid storage system. For example, flash-based SSDs are fast but more expensive (compared to HDDs). Many

Device	Performance	Price	Capacity	Endurance	Power
Tape	*	*	****	*****	*
HDD	**	***	***	****	***
SMR	**	**	****	****	***
SSD	****	****	***	**	*
PCM	****	*****	*	****	*
DRAM	*****	*****	**	****	*
Other SCMs	N/A	N/A	N/A	N/A	N/A

Table 10.1: Summary of storage devices. We use numbers of "*" to indicate the degree. For example, "*" means very low or very small; "**" means low or small; "***" means medium, "****" means high or large; and "*****" means very high or very large.

studies show that SSDs can be utilized either as a caching tier or a persistent tier for a more efficient storage system design [11, 37, 103, 112, 129]. However, SSDs suffer from endurance and unpredictable garbage collections.

Emerging PCMs [25] are even faster, more durable, and are byte addressable; but PCMs have much smaller capacity per dollar. The trade-off between placing PCMs and SSDs is an interesting question to explore.

To a lower tier, Shingled Magnetic Recording (SMR) [20] disks are emerging to further lower down the cost of HDDs. SMR disks require more sophisticated drive controllers or file systems to make them work: they offer more space than HDDs, but require writing data sequentially for efficiency, which requires an SSD-like FTL and garbage-collection support. Shingling is a technique that is applicable to other future magnetic recording devices (e.g., BPR [16] and HAMR [55]). For this future work, we can use the SMR prototypes we have through a unique collaboration with industry.

We provide a summary table for new-old storage devices in Table 10.1. One thing worth mentioning is that the prices of the various storage devices can vary depending on the technology, market, and economics. It makes our work more useful in the future.

However, there is no one-to-all solution combining these devices to build hybrid drive storage systems. Placing different storage devices in different tiers, together with different policies (e.g., data movement throttling, I/O management, meta-data management), will produce vary different results and trace-offs in terms of performance, energy, and endurance of the hybrid drive storage system. We would be targeting SSDs, PCMs, and SMRs toward providing cost-effective I/O services of hybrid storage systems.

Specifically, it would be interesting to answer the following two questions:

1. What if we utilize the PCM instead of the SSD as the front tier?

2. What if we utilize the SMR instead of the HDD as the back-end tier?

10.5 Provide Control Support at the CPU Level

This section further details future work on providing control support at the CPU level as well. Depending on the observed results, our analysis can help build more intelligent hybrid drive stor-

age systems. How to automate the controllable parameters is something we discussed earlier in Section 10.1.

We have another project, named “CPUIDLE”, that aims to let a user control the low-power states of individual CPU cores [99]. Therefore, we can extend the reach of our storage system to provide controls not only at the block level, but also at the CPU and core level. Therefore, there will have more controllable parameters to explore in terms of tackling the trade-offs between the CPU performance and the power consumption.

It would be interesting to merge the CPUIDLE kernel module with our GreenDM module, and perform more experiments (e.g., turning CPU cores on/off in accordance with workloads) and do more analysis toward the trade-offs among performance, energy, and endurance of the whole systems. Depending on the observed results, our analysis can help build more intelligent hybrid drive storage systems, considering the performance, energy efficiency, and endurance of the entire systems.

Chapter 11

Conclusion

There are trade-offs among performance, energy, and device endurance for storage systems. Designs optimized for one dimension or workload often suffer in another. Therefore, it is important to study the trade-offs so as to be able to adapt the system to workloads. As different types of drives have different traits, hybrid drives are studied more closely. However, previous hybrids are often designed for high throughput, efficient energy consumption, or improving endurance—leaving empirical study on the trade-offs being unexplored. Past endurance studies also lack a concrete model and metric to help study the trade-offs. Lastly, previous designs are often based on inflexible policies that cannot adapt easily to changing conditions.

Our previous study has looked at the power consumption issue in enterprise-scale backup storage systems in Chapter 4 to gain us more domain knowledge on power and energy in the field. Besides, our previous study in Chapter 5 on the trade-offs between performance and energy has given us the understanding of variations for both the performance and energy with varying software and hardware conditions. We presented GreenDM in Chapter 6 as a versatile tiering hybrid drive to study empirically the trade-offs among performance, energy, and endurance. We also provided several interesting observations regarding the associated cost dimension of GreenDM in Chapter 7. We further presented the follow-up work on our caching system based on the same hardware and similar software setup in Chapter 8. We also presented additional work on evaluating both the tiering and caching systems with a different capacity ratio of SSD over total in Chapter 9.

To conclude with several interesting future research topics (see Chapter 10). First, it will be interesting to provide automated control knobs for the user to trade-off performance, energy efficiency, and endurance. Second, one could extend the two-tier system to three tiers and explore more tiering policies. Third, it would be useful to add security as a forth dimension to further explore the trade-offs. Forth, one could experiment with different storage devices and policies in the future, and help build more efficient storage systems to achieve the high performance with the minimum cost. Last but not least, it would be interesting to provide control support at the CPU level as well to further justify the trade-offs among performance, energy, and endurance.

Bibliography

[1] 80 PLUS Certified Power Supplies and Manufacturers. `www.plugloadsolutions.com/80PlusPowerSupplies.aspx`.

[2] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[3] H. Amur, J. Cipar, V. Gupta, G. R. Ganger, M. A. Kozuch, and K. Schwan. Robust and Flexible Power-Proportional Storage. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, 2010.

[4] Taneja Group Technology Analysts. The state of the core — engineering the enterprise storage infrastructure with the ibm ds8000. Technical report, IBM, 2010.

[5] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan. FAWN: A Fast Array of Wimpy Nodes. In *Proceedings of the 22nd ACM Symposium on Operating Systems Principles (SOSP '2009)*. ACM SIGOPS, October 2009.

[6] J. Axboe. CFQ IO Scheduler, 2007. `http://mirror.linux.org.au/pub/linux.conf.au/2007/video/talks/123.ogg`.

[7] L. N. Bairavasundaram, G. R. Goodson, B. Schroeder, A. C. Arpaci-dusseau, and R. H. Arpaci-dusseau. An Analysis of Data Corruption in the Storage Stack. In *Proceedings of the Sixth USENIX Conference on File and Storage Technologies (FAST '08)*, San Jose, CA, February 2008. USENIX Association.

[8] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation algorithms for deadline-tsp and vehicle routing with time-windows. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 166–174. ACM, 2004.

[9] L. A. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *Computer*, 40:33–37, December 2007.

[10] L. A. Barroso and U. Hölzle. The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines. *Synthesis Lectures on Computer Architecture*, 4(1):1–108, 2009.

[11] Bcache. `http://bcache.evilpiepirate.org/`.

[12] F. Bellosa. The benefits of event: driven energy accounting in power-sensitive systems. In *Proceedings of the 9th workshop on ACM SIGOPS European workshop*, pages 37–42, 2000.

[13] W.L. Bircher and L.K. John. Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events. In *Proceedings of the 2007 IEEE International Symposium on Performance Analysis of Systems and Software*, pages 158–168, 2007.

[14] T. Bisson, S. A. Brandt, and D. D.E. Long. A Hybrid Disk-Aware Spin-Down Algorithm with I/O Subsystem Support. In *In Proceedings of the 26th IEEE International Performance, Computing and Communications Conference*, 2007.

[15] P. Bohrer, E. N. Elnozahy, T. Keller, M. Kistler, and L. C. Mcdowell. The case for Power Management in Web Servers, 2002. `www.research.ibm.com/people/l/lefurgy/Publications/pac2002.pdf`.

[16] Patterned Media. `http://en.wikipedia.org/wiki/Patterned_media`.

[17] A. Brown and M. Seltzer. Operating System Benchmarking in the Wake of Lmbench: A Case Study of the Performance of NetBSD on the Intel x86 Architecture. In *Proceedings of the 1997 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 214–224. ACM Press, June 1997.

[18] Bzip2 Wikipedia Documentation. `http://en.wikipedia.org/wiki/Bzip2`.

[19] A. Carroll and G. Heiser. An Analysis of Power Consumption in a Smartphone. In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, Boston, MA, USA, 2010.

[20] Y. Cassuto, M. A. A. Sanvido, C. Guyot, D. R. Hall, and Z. Z. B. Indirection systems for shingled-recording disk drives. In *Proceedings of the International IEEE Symposium on Mass Storage Systems and Technologies (MSST)*, Incline Village, Nevada, May 2010. IEEE.

[21] CFDR - the computer failure data repository. `cfdr.usenix.org`.

[22] J. Chang, J. Meza, P. Ranganathan, C. Bash, and A. Shah. Green Server Design: Beyond Operational Energy to Sustainability. In *Proceedings of the 2010 International Conference on Power Aware Computing and Systems*, HotPower'10, 2010.

[23] F. Chen, D. A. Koufaty, and X. Zhang. Hystor: Making the Best Use of Solid State Drives in High Performance Storage Systems. In *Proceedings of the International Conference on Supercomputing*, pages 22–32, 2011.

[24] D. Colarelli and D. Grunwald. Massive Arrays of Idle Disks for Storage Archives. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–11, 2002.

[25] Jeremy Condit, Edmund B. Nightingale, Christopher Frost, Engin Ipek, Benjamin Lee, Doug Burger, and Derrick Coetzee. Better i/o through byte-addressable, persistent memory. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, SOSP '09, pages 133–146, New York, NY, USA, 2009. ACM.

[26] A. Coskun, R. Strong, D. Tullsen, and T. S. Rosing. Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors. In *Proceedings of the 2009 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, 2009.

[27] D. C. Montgomery. *Engineering Statistics*. Wiley, 3 edition, 2004.

[28] Data Domain Boost Software, EMC Corporation, 2012. `www.datadomain.com/products/dd-boost.html`.

[29] V. Delaluz, A. Sivasubramaniam, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin. Scheduler-Based DRAM Energy Management. In *Proceedings of the 39th annual Design Automation Conference*, pages 697–702, New York, USA, 2002.

[30] Dell Compellent Flash Optimized Solutions. `http://www.dell.com/us/business/p/dell-compellent-flash-optimized/pd?~ck=anav`.

[31] G. Dhiman and T. Rosing. System-level Power Management using Online Learning. *IEEE Transaction on Computer-Aided Design of Integrated Circuits Systems*, 28(5):676–689, 2009.

[32] J. G. Elerath. Server Class Disk Drives: How Reliable Are They. In *IEEE Reliability and Maintainability Symposium*, pages 151–156, 2004.

[33] D. Essary and A. Amer. Predictive Data Grouping: Defining the Bounds of Energy and Latency Reduction through Predictive Data Grouping and Replication. *ACM Transactions on Storage (TOS)*, 4(1):1–23, May 2008.

[34] Fast Fourier Transform. `http://en.wikipedia.org/wiki/Fast_Fourier_transform`.

[35] Filebench. `http://filebench.sf.net`.

[36] Fiu srcmap trace repository. `http://iotta.snia.org/traces/414`.

[37] Flashcache. `https://github.com/facebook/flashcache/`.

[38] David Floyer. Enterprise Flash Drive Cost and Technology Projections, 2009. `http://wikibon.org/wiki/v/Enterprise_Flash_Drive_Cost_and_Technology_Projections`.

[39] Fluke 345 Power Quality Clamp Meter. `www.fluke.com/fluke/caen/products/categorypqttop.htm`.

[40] J. L. Gailly. GNU Zip. `www.gnu.org/software/gzip/gzip.html`, 2000.

[41] J. Gantz and D. Reinsel. The digital universe decade - are you ready? `www.emc.com/digital_universe`, May 2010.

[42] Gartner, Inc. Server Storage and RAID Worldwide. Technical report, Gartner Group/Dataquest, 1999. www.gartner.com.

[43] Gartner, Inc. Desktop Total Cost of Ownership: 2013 Update. Technical report, Gartner Group/Dataquest, 2013. www.gartner.com.

[44] Gartner, Inc. Notebook Total Cost of Ownership: 2013 Update. Technical report, Gartner Group/Dataquest, 2013. www.gartner.com.

[45] K. M. Greenan, J. S. Plank, and J. J. Wylie. Mean Time to Meaningless: MTTDL, Markov models, and Storage System Reliability. In *HotStorage '10: Proceedings of the 2nd USENIX Workshop on Hot Topics in Storage*, 2010.

[46] D. Grunwald, C. B. Morrey III, P. Levis, M. Neufeld, and K. I. Farkas. Policies for dynamic clock scheduling. In *Proceedings of the 4th Symposium on Operating System Design & Implementation*, San Diego, CA, 2000.

[47] B. Guenter, N. Jain, and C. Williams. Managing Cost, Performance, and Reliability Tradeoffs for Energy-Aware Server Provisioning. In *INFOCOM 2011. 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies*, pages 1332–1340, 2011.

[48] J. Guerra, W. Belluomini, J. Glider, K. Gupta, and H. Pucha. Energy Proportionality for Storage: Impact and Feasibility. *ACM SIGOPS Operating Systems Review*, pages 35 – 39, 2010.

[49] J. Guerra, H. Pucha, J. Glider, W. Belluomini, and R. Rangaswami. Cost Effective Storage Using Extent Based Dynamic Tiering. In *USENIX FAST*, 2011.

[50] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke. DRPM: Dynamic Speed Control for Power Management in Server Class Disks. In *Proceedings of the 30th International Symposium on Computer Architecture*, pages 169–179, San Diego, Californai, USA, 2003.

[51] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke. DRPM: Dynamic Speed Control for Power Management in Server Class Disks. In *Proceedings of the 30th Annual International Symposium on Computer Architecture*, pages 169–181, 2003.

[52] H. Kim and S. Seshadri and C. L. Dickey and L. Chiu. Evaluating Phase Change Memory for Enterprise Storage Systems: A Study of Caching and Tiering Approaches. In *Proceedings of the 12th USENIX Conference on File and Storage Technologies*, pages 33–45, Berkeley, CA, 2014. USENIX.

[53] H. Shim and J. Kim and S. Maeng. BEST: Best-effort Energy Saving Techniques for NAND Flash-based Hybrid Storage. *IEEE Trans. Consumer Electronics*, pages 841–848, 2012.

[54] G. Hamerly and C. Elkan. Bayesian Approaches to Failure Prediction for Disk Drives. In *In Proceedings of the Eighteenth International Conference on Machine Learning*, pages 202–209. Morgan Kaufmann, 2001.

[55] Heat-Assisted Magnetic Recording. http://en.wikipedia.org/wiki/Heat-assisted_magnetic_recording.

[56] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: The next generation. In *IEEE Real-Time Systems Symposium*, pages 56–65, 1995.

[57] Hill Climbing. https://en.wikipedia.org/wiki/Hill_climbing.

[58] Hitachi Deskstar 7K2000. www.hitachigst.com/deskstar-7k2000.

[59] M. Hofri. Disk Scheduling: FCFS vs. SSTF revisited. *Communication of the ACM*, 23(11), November 1980.

[60] H. Huang, W. Hung, and K. Shin. FS2: Dynamic Data Replication in Free Disk Space for Improving Disk Performance and Energy Consumption. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP '05)*, pages 263–276, Brighton, UK, October 2005. ACM Press.

[61] A. A. Hwang, I. A. Stefanovici, and B. Schroeder. Cosmic Rays Don't Strike Twice: Understanding the Nature of DRAM Errors and the Implications for System Design. In *ASPLOS XVII Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 111–122. ACM, 2012.

[62] C. Isci, G. Contreras, and M. Martonosi. Live, runtime phase monitoring and prediction on real systems with application to dynamic power management. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2006.

[63] W. Jiang, C. Hu, Y. Zhou, and A. Kanevsky. Are disks the dominant contributor for storage failures? a comprehensive study of storage subsystem failure characteristics. In *Proceedings of the Sixth USENIX Conference on File and Storage Technologies (FAST '08)*, pages 111–125, San Jose, CA, February 2008. USENIX Association.

[64] N. Joukov and J. Sipek. GreenFS: Making Enterprise Computers Greener by Protecting Them Better. In *Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems 2008 (EuroSys 2008)*, Glasgow, Scotland, April 2008. ACM.

[65] N. Joukov, A. Traeger, R. Iyer, C. P. Wright, and E. Zadok. Operating System Profiling via Latency Analysis. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI 2006)*, pages 89–102, Seattle, WA, November 2006. ACM SIGOPS.

[66] Asim Kadav, Mahesh Balakrishnan, Vijayan Prabhakaran, and Dahlia Malkhi. Differential raid: Rethinking raid for ssd reliability. In *HotStorage '09: Proceedings of the 1st Workshop on Hot Topics in Storage*. ACM, 2009.

[67] R. T. Kaushik and M. Bhandarkar. GreenHDFS: Towards An Energy-Conserving, Storage-Efficient, Hybrid Hadoop Compute Cluster. In *Proceedings of the 2010 International Conference on Power Aware Computing and Systems*, HotPower'10, 2010.

[68] Y. Kim, A. Gupta, B. Urgaonkar, P. Berman, and A. Sivasubramaniam. HybridStore: A Cost-Efficient, High-Performance Storage System Combining SSDs and HDDs. In *IEEE MASCOTS*, 2011.

[69] R. King. Disk Arm Movement Anticipation of Future Requests. *ACM Transactions on Computer Systems*, 8(3):214–229, 1990.

[70] I. Koltsidas and S. D. Viglas. Flashing up the Storage Layer. *Proceedings of the VLDB Endowment*, 1:514–525, 2008.

[71] J. G. Koomey. Growth in Data Center Electricity Use 2005 to 2010. Technical report, Standord University, 2011. `www.koomey.com`.

[72] R. Kothiyal, V. Tarasov, P. Sehgal, and E. Zadok. Energy and Performance Evaluation of Lossless File Data Compression on Server Systems. In *Proceedings of the Second ACM Israeli Experimental Systems Conference (SYSTOR '09)*, Haifa, Israel, May 2009. ACM.

[73] B. Laliberte. Automate and optimize a tiered storage environment fast. Technical report, EMC, 2009.

[74] S. Lee, T. Kim, K. Kim, and J. Kim. Lifetime Management of Flash-Based SSDs Using Recovery-Aware Dynamic Throttling. In *Proceedings of the Tenth USENIX Conference on File and Storage Technologies (FAST '12)*, San Jose, CA, February 2012. USENIX Association.

[75] S. Li and A. Belay. cpuidle — Do nothing, efficiently... In *Proceedings of the Linux Symposium*, volume 2, Ottawa, Ontario, Canada, 2007.

[76] Z. Li, A. Desai, C. Bhatt, and E. Zadok. vATM: vSphere Adaptive Task Management. In *In Proceedings of the Seventh International Workshop on Feedback Computing (FC'12)*, September 2012.

[77] Z. Li, K. M. Greenan, A. W. Leung, and E. Zadok. Power Consumption in Enterprise-Scale Backup Storage Systems. In *Proceedings of the Tenth USENIX Conference on File and Storage Technologies (FAST '12)*, San Jose, CA, February 2012. USENIX Association.

[78] Z. Li, R. Grosu, K. Muppalla, S. A. Smolka, S. D. Stoller, and E. Zadok. Model discovery for energy-aware computing systems: An experimental evaluation. In *Proceedings of the 1st Workshop on Energy Consumption and Reliability of Storage Systems (ERSS'11)*, Orlando, FL, July 2011.

[79] Z. Li, R. Grosu, P. Sehgal, S. A. Smolka, S. D. Stoller, and E. Zadok. On the Energy Consumption and Performance of Systems Software. In *Proceedings of the 4th Israeli Experimental Systems Conference (ACM SYSTOR '11)*, Haifa, Israel, May/June 2011. ACM.

[80] L. Ljung. *System Identification (2nd ed.): Theory for the User*. Prentice Hall, 1999.

[81] J. R. Lorch. A Complete Picture of the Energy Consumption of a Portable Computer. Master's thesis, University of California at Berkeley, 1995. `http://research.microsoft.com/users/lorch/papers/masters.ps`.

[82] Y. Lu, J. Shu, and W. Zheng. Extending the Lifetime of Flash-based Storage through Reducing Write Amplification from File Systems. In *In Proceedings of the 11th USENIX Symposium on File and Storage Technologies (FAST '13)*, 2013.

[83] T. Luo, R. Lee, M. Mesnier, F. Chen, and X. Zhang. hStorage-DB: Heterogeneity-Aware Data Management to Exploit the Full Capability of Hybrid Storage Systems. *Proceedings of the VLDB Endowment*, pages 1076–1087, 2012.

[84] M. Jung and M. Kandemir. Revisiting Widely Held SSD Expectations and Rethinking System-Level Implications. In *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '13, pages 203–216, New York, NY, USA, 2013. ACM.

[85] M. Wei and L M. Grupp and F. E. Spada and S. Swanson. Reliably Erasing Data from Flash-Based Solid State Drives. In *Proceedings of the 9th USENIX Conference on File and Stroage Technologies*, FAST '11, Berkeley, CA, USA, 2011. USENIX Association.

[86] Machine Learning. `http://en.wikipedia.org/wiki/Machine_learning`.

[87] Markov Chain. `http://en.wikipedia.org/wiki/Markov_chain`.

[88] S. M Martin, K. Flautner, T. N. Mudge, and D. Blaauw. Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. In *Proceedings of the 2002 IEEE/ACM International Conference on Computer-Aided Design*, pages 721–725, 2002.

[89] A. Miyoshi, C. Lefurgy, E. V. Hensbergen, R. Rajamony, and R. Rajkumar. Critical power slope: understanding the runtime effects of frequency scaling. In *Proceedings of the 16th International Conference on Supercomputing (ICS '02)*, pages 35–44, 2002.

[90] J. F. Murray, G. F. Hughes, and D. Schuurmans. Machine Learning Methods for Predicting Failures in Hard Drives: A Multiple-Instance Application. *Journal of Machine Learning research*, page 816, 2005.

[91] D. Narayanan, A. Donnelly, and A. Rowstron. Write off-loading: Practical Power Management for Enterprise Storage. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST 2008)*, 2008.

[92] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron. Migrating server storage to ssds: analysis of tradeoffs. In *EuroSys '09: Proceedings of the 4th ACM European conference on Computer systems*, pages 145–158, New York, NY, USA, 2009. ACM.

[93] NetApp Flash Pool. `http://www.netapp.com/us/products/platform-os/flashpool.aspx`.

[94] Nimble's Hybrid Storage Architecture. `www.nimblestorage.com/products/architecture.php`.

[95] M.F.X.J. Oberhumer. lzop data compression utility. `www.lzop.org/`.

[96] Symantec OpenStorage, Symantec Corporation, 2012. www.symantec.com/theme.jsp?themeid=openstorage.

[97] P. Desnoyers. What Systems Researchers Need to Know about NAND Flash. In *HotStorage '13: Proceedings of the 5th USENIX Workshop on Hot Topics in Storage*, 2013.

[98] B. Palanisamy, A. Singh, L. Liu, and B. Langston. Cura: A Cost-Optimized Model for MapReduce in a Cloud. In *Proc. of 27th IEEE International Parallel and Distributed Processing Symposium*, 2013.

[99] M. Palmur, Z. Li, and E. Zadok. Cpuidle from user space. Technical Report FSL-13-05, Computer Science Department, Stony Brook University, December 2013.

[100] R. Panabaker. Hybrid Hard Disk and Ready-Drive Technology: Improving Performance and Power for Windows Vista Mobile PCs, 2006. http://www.microsoft.com/whdc/winhec/pres06.mspx.

[101] E. Pinheiro and R. Bianchini. Energy Conservation Techniques for Disk Array-Based Servers. In *Proceedings of the 18th International Conference on Supercomputing (ICS 2004)*, pages 68–78, 2004.

[102] E. Pinheiro, W. Weber, and L. A. Barroso. Failure trends in a large disk drive population. In *Proceedings of the Fifth USENIX Conference on File and Storage Technologies (FAST '07)*, pages 17–28, San Jose, CA, February 2007. USENIX Association.

[103] T. Pritchett and M. Thottethodi. SieveStore: A Highly-Selective, Ensemble-level Disk Cache for Cost-Performance. In *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ISCA '10, 2010.

[104] R. Freitas. Storage Class Memory: Technology, Systems and Applications. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 985–986, 2009.

[105] R. Jain. *The Art of Computer System Performance Analysis*. Wiley, 1991.

[106] RAID 6 Equation Calculator. http://raideqn.netapp.com/.

[107] David S. H. Rosenthal, Daniel C. Rosenthal, Ethan L. Miller, Ian F. Adams, Mark W. Storer, and Erez Zadok. The economics of long-term digital storage. In *The Memory of the World in the Digital age: Digitization and Preservation*. United Nations Educational, Scientific and Cultural Organization (UNESCO), September 2012.

[108] A. Sagahyroon. Power Consumption Breakdown on a Modern Laptop. In *Proceedings of the 2004 Workshop on Power-Aware Computer Systems*, pages 165–180, Portland, OR, 2004.

[109] A. Sagahyroon. Analysis of dynamic power management on multi-core processors. In *Proceedings of the International Symposium on Circuits and Systems*, pages 1721–1724, 2006.

[110] A. Sagahyroon. Power Consumption in Handheld Computers. In *Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems*, pages 1721–1724, Singapore, 2006.

[111] R. Sarikaya, C. Isci, and A. Buyuktosunoglu. Runtime workload behavior prediction using statistical metric modeling with application to dynamic power management. In *In IEEE International Symposium on Workload Characterization*, 2010.

[112] M. Saxena, M. M. Swift, and Y. Zhang. FlashTier: a Lightweight, Consistent and Durable Storage Cache. In *Proceedings of the 7th ACM European Conference on Computer Systems*, EuroSys '12, pages 267–280, 2012.

[113] B. Schroeder and G. A.Gibson. A Large-Scale Study of Failures in High-Performance Computing Systems. In *Proceedings of the International Conference on Dependable Systems and Networks*, DSN '06, pages 249–258. IEEE Computer Society, 2006.

[114] B. Schroeder, S. Damouras, and P. Gill. Understanding Latent Sector Errors and How to Protect against Them. In *FAST'10: Proceedings of the 8th USENIX Conference on File and Storage Technologies*. ACM, 2010.

[115] B. Schroeder and G. A. Gibson. Disk failures in the real world: What does an mttf of 1,000,000 hours mean to you? In *Proceedings of the Fifth USENIX Conference on File and Storage Technologies (FAST '07)*, pages 1–16, San Jose, CA, February 2007. USENIX Association.

[116] B. Schroeder and G. A. Gibson. Understanding Failures in Petascale Computers. In *Proceedings of the Scientific Discovery through Advanced Computing*, SciDAC'07, 2007.

[117] 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*, SIGMETRICS '09, pages 193–204. ACM, 2009.

[118] G. Schulz. Storage Industry Trends and IT Infrastructure Resource Management (IRM), 2007. `www.storageio.com/DownloadItems/CMG/MSP_CMG_May03_2007.pdf`.

[119] P. Sehgal, V. Tarasov, and E. Zadok. Evaluating Ferformance and Energy in File System Server Workloads. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*, pages 253–266, San Jose, CA, February 2010. USENIX Association.

[120] G. Soundararajan, V. Prabhakaran, M. Balakrishnan, and T. Wobber. Extending SSD Lifetimes with Disk-Based Write Caches. In *FAST'10: Proceedings of the 8th USENIX Conference on File and Storage Technologies*, Berkeley, CA, USA, February 2010. USENIX Association.

[121] V. Srinivasan, G. R. Shenoy, S. Vaddagiri, and D. Sarma. Energy-aware task and interrupt management in linux. In *Ottawa Linux Symposium*, July 2008.

[122] M. W. Storer, K. M. Greenan, E. L. Miller, and K. Voruganti. Pergamum: Replacing Tape with Energy Efficient, Reliable, Disk-based Archival Storage. In *Proceedings of the Sixth USENIX Conference on File and Storage Technologies (FAST '08)*, San Jose, CA, February 2008. USENIX Association.

[123] J. D. Strunk. Hybrid Aggregates: Combining SSDs and HDDs in a Single Storage Pool. *SIGOPS Oper. Syst. Rev.*, pages 50–56, 2012.

[124] E. L. Sueur and G. Heiser. Slow Down or Sleep, That Is The Question. In *Proceedings of the 2011 USENIX Annual Technical Conference*, Portland, Oregon, USA, 2011.

[125] Y. Tan and X. Gu. On Predictability of System Anomalies in Real World. In *Proceedings of the 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, MASCOTS '10, pages 133–140. IEEE Computer Society, 2010.

[126] V. Tarasov, S. Bhanage, E. Zadok, and M. Seltzer. Benchmarking File System Benchmarking: It *IS* Rocket Science. In *Proceedings of HotOS XIII:The 13th USENIX Workshop on Hot Topics in Operating Systems*, Napa, CA, May 2011.

[127] Total Cost of Ownership. `http://en.wikipedia.org/wiki/Total_cost_of_ownership`.

[128] E. Thereska, A. Donnelly, and D. Narayanan. Sierra: a Power-proportional, Distributed Storage System. In *Proceedings of EuroSys 2011*, 2011.

[129] A Tiered Block Device. `http://sourceforge.net/projects/tier/`.

[130] UMass trace repository. `http://traces.cs.umass.edu`.

[131] V. Mohan and T. Siddiqua and S. Gurumurthi and M. R. Stan. How I Learned to Stop Worrying and Love Flash Endurance. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Storage and File Systems*, HotStorage'10, 2010.

[132] A. Verma, R. Koller, L. Useche, and R. Rangaswami. SRCMap: Energy Proportional Storage Using Dynamic Consolidation. In *Proceedings of the 8th USENIX Conference on File and Storage Technologies*, FAST'10, 2010.

[133] Virtual Flash Tech Preview. `http://blogs.vmware.com/vsphere/2012/12/virtual-flash-vflash-tech-preview.html`.

[134] Tintri VMStore. `www.tintri.com/resources/videos/introduction-to-tintri/`.

[135] G. Wallace, F. Douglis, H. Qian, P. Shilane, S. Smaldone, M. Chamness, and W. Hsu. Characteristics of Backup Workloads in Production Systems. In *Proceedings of the Tenth USENIX Conference on File and Storage Technologies (FAST '12)*, San Jose, CA, February 2012. USENIX Association.

[136] Watts up? PRO ES Power Meter. `www.wattsupmeters.com/secure/products.php`.

[137] Wavelet. `http://en.wikipedia.org/wiki/Wavelet`.

[138] WD Blue. `http://www.wd.com/en/products/products.aspx?id=800#tab11`.

[139] C. Weddle, M. Oldham, J. Qian, A. A. Wang, P. Reiher, and G. Kuenning. PARAID: a Gear-Shifting Power-Aware RAID. In *Proceedings of the Fifth USENIX Conference on File and Storage Technologies (FAST '07)*, pages 245–260, San Jose, CA, February 2007. USENIX Association.

[140] Device Mapper. `http://en.wikipedia.org/wiki/Device_mapper`.

[141] Fusion Drive. `http://en.wikipedia.org/wiki/Fusion_Drive`.

[142] Gamma Distribution. `http://en.wikipedia.org/wiki/Gamma_distribution`.

[143] Weibull Distribution. `http://en.wikipedia.org/wiki/Weibull_distribution`.

[144] A. Wildani and E. Miller. Semantic Data Placement for Power Management in Archival Storage. In *PDSW 2010*, New Orleans, LA, USA, 2010. ACM.

[145] A. W. Wilson. Operation and implementation of random variables in Filebench.

[146] C. P. Wright, N. Joukov, D. Kulkarni, Y. Miretskiy, and E. Zadok. Auto-pilot: A Platform for System Software Benchmarking. In *Proceedings of the Annual USENIX Technical Conference, FREENIX Track*, pages 175–187, Anaheim, CA, April 2005. USENIX Association.

[147] T. Xie and Y. Sun. PEARL: Performance, Energy, and Reliability Balanced Dynamic Data Redistribution for Next Generation Disk Arrays. In *IEEE MASCOTS*, 2008.

[148] J Yang and F. Sun. A Comprehensive Review of Hard-Disk Drive Reliability. In *Annual Reliability and Maintainability Symposium*, 1999.

[149] Y. Yu, D. Shin, H. Eom, and H. Yeom. NCQ vs I/O scheduler: preventing unexpected misbehaviors. *ACM Transaction on Storage*, 6(1), March 2010.

[150] J. Yue, Y. Zhu, Z. Cai, and L. Lin. Energy and thermal aware buffer cache replacement algorithm. In *The 26th IEEE Symposium on Massive Storage Systems and Technologies*, 2010.

[151] ZCAV. `http://en.wikipedia.org/wiki/Zone_bit_recording`.

[152] B. Zhu, K. Li, and H. Patterson. Avoiding the Disk Bottleneck in the Data Domain Deduplication File System. In *Proceedings of the Sixth USENIX Conference on File and Storage Technologies (FAST '08)*, San Jose, California, USA, 2008.

[153] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes. Hibernator: Helping Disk Arrays Sleep Through the Winter. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP '05)*, pages 177–190, Brighton, UK, October 2005. ACM Press.

[154] Q. Zhu, F. M. David, C. F. Devaraj, Z. Li, Y. Zhou, and P. Cao. Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management. In *Proceedings of the 10th International Symposium on High-Performance Computer Architecture*, pages 118–129, 2004.

[155] X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, P. Padala, and K. Shin. What does control theory bring to systems research? *SIGOPS Oper. Syst. Rev.*, pages 62–69, 2009.

[156] Y. Zhu and F. Mueller. Feedback EDF Scheduling Exploiting Dynamic Voltage Scaling. In *RTAS '04: Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 33 – 63, Washington, DC, USA, 2004. IEEE Computer Society.