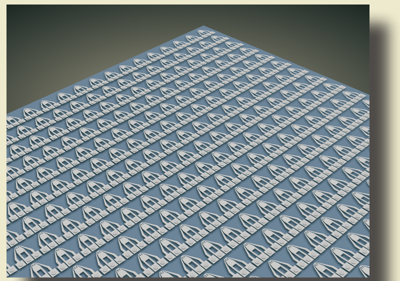


MEMS-BASED STORAGE DEVICES

Integration in Energy-Constrained Mobile Systems

In the digital era, the amount of data a person stores has multiplied significantly. The growth for storage capacity is increasing by about 30% - 60% per year. People keep track of their personal data for every event in their life, while new regulations stipulate storing business information for long periods of time.

This dissertation puts in the hands of the reader a four-dimensional investigation of a new class of storage devices, called MEMS-based storage devices. The research reported on enhances the energy-efficiency, reduces the cost, increases the performance, and extends the lifetime of this class of devices.

An emperical comparison to Flash memory is carried out in the frame of mobile battery-powered devices. Optimization results support the big potential promised by MEMS-based storage devices, demonstrating their ability to take off in the storage-demanding energy-aware digital era.

Mohammed G. Khatib received his Master's degree in Computer Science from the Technical University of Braunschweig in 2004, where he worked as a researcher from 2002 through 2004. He visited the Storage Systems Research Center of the University of California at Santa Cruz in Spring of 2008.

Mohammed received several awards of academic excellence, and received a number of travel grants. His research interests include computer architecture and storage systems.

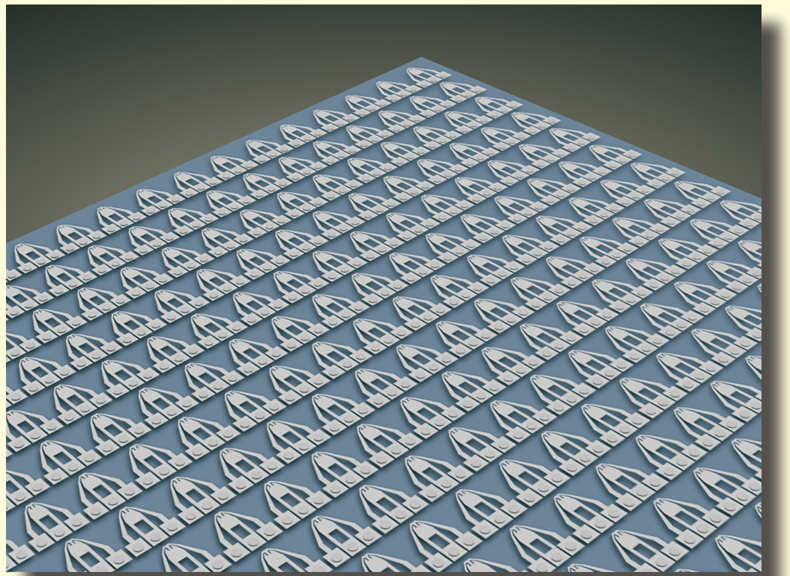
Cover desing by the author
Cover picture courtesy from
IBM Research Laboratories, Zurich

ISBN: 978-90-365-2847-4

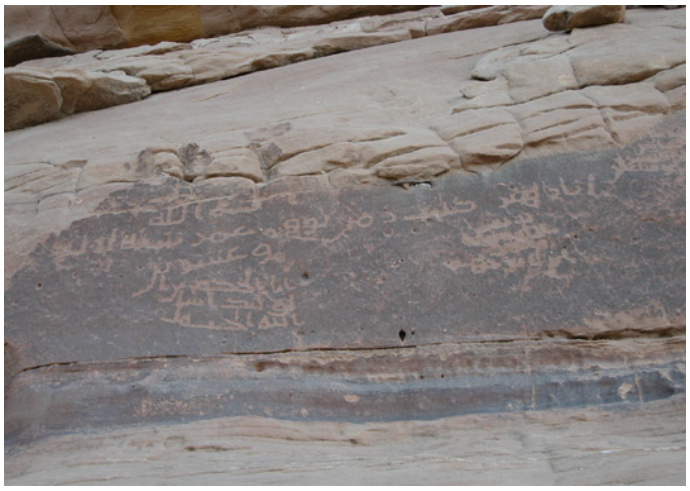
MEMS-Based Storage Devices   Mohammed G. Khatib

MEMS-BASED STORAGE DEVICES

Integration in Energy-Constrained Mobile Systems



Mohammed G. Khatib



The opening-page photograph shows the earliest dated Islamic inscription written in Arabic in the Hijazi script. It dates back to the 24 AH (644 CE). The remarkable characteristic is the presence of dotted consonants, which was not the case in the older scripts. The inscription examples the existence of data recording for a very long time. The inscription translates from top to bottom as follows: In the name of God, I, Zuhayr, wrote [this] at the time 'Umar died in the year four, and twenty (i.e., 24 AH). The photo is adopted from the Discovery Channel at `http://dsc.discovery.com/news/2008/11/18/islamic-inscription.html`. More information can be obtained at `http://www.islamic-awareness.org/History/Islam/Inscriptions/kuficsaud.html`

MEMS-Based Storage Devices

Integration in Energy-Constrained Mobile Systems

Dissertation

Mohammed G. Khatib

The research in this dissertation was supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of Economic Affairs under project number TES.06369.

Computer Architecture for Embedded Systems group
Distributed and Embedded Security group
The Faculty of Electrical Engineering,
Mathematics and Computer Science
P. O. Box 217
7500 AE Enschede
The Netherlands

ISSN: 1381-3617, CTIT Ph.D.-thesis series No. 09-145
Centre for Telematics and Information Technology
P.O.Box 217
7500 AE Enschede
The Netherlands

IPA Ph.D.-thesis series No. 2009-12
The work in this thesis has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics).

ISBN 978-90-365-2847-4
DOI 10.3990./1.9789036528474

MEMS-Based Storage Devices
Integration in Energy-Constrained Mobile Systems

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof. dr. H. Brinksma,
on account of the decision of the graduation committee,
to be publicly defended
on Thursday, June 11, 2009 at 13:15 hrs

by

Mohammed Ghiath Khatib

born on 4 June 1979,
in Aleppo, Syrian Arab Republic

The dissertation is approved by:

Prof. dr.	P. H. Hartel	University of Twente (promoter)
Dr. ir.	L. Abelmann	University of Twente (co-promoter)

ABSTRACT

The digital era in which we are living today requires our increasing awareness of energy efficiency to reduce the negative effects on our lovely environment. We, people, are increasingly dealing with digital contents to facilitate our deals, which increases the demand for larger storage capacities than ever before.

The environmental considerations and the data explosion worldwide are calling for green and ultrahigh-density storage technologies. A storage technology, based on Micro-Electro-Mechanical Systems (MEMS), promises to deliver green and high-capacity storage systems. Storage densities up to 4 Tb/in^2 have already been demonstrated. With such a technology, a storage device with a capacity of 1 Tb can be mounted in a package smaller than a thumbnail, dissipating one Watt of power.

Disruptive technologies take, however, a significant amount of time to materialize into commercial products. One key reason for this delay is the difficulty of integrating and adopting new technologies. Integration, in a broad sense, involves the investigation of roles a new technology can play, and solutions to its impediments. Early solutions to the integration problem help to reduce the time to market, and most likely contribute to the success of the technology.

Flash memory, for instance, was invented in the eighties. The large demand for Flash in mobile systems has drawn the attention of researchers to investigate Flash. Just recently, researchers started looking into ways to construct storage systems based on Flash that are reliable, have low latency, and consume little energy. Flash memory has not found its way to enterprise storage yet, whereas their kin, hard disk drives, are sitting there wasting a significant amount of energy. That kind of late response to Flash is costing data centers millions of dollars every day in energy and cooling cost.

We would like to avoid such a late response for MEMS-based storage devices by being proactive in how we can get this family of devices successfully integrated as early as possible. Like any other technology, MEMS-based storage demands optimization, and has challenges that need to be tackled. In this work, we optimize MEMS-based storage, tailor it to mobile battery-powered systems, and compare it to Flash memory and Hybrid (Disk–Flash) storage.

The research of this dissertation looks mainly into the energy and cost aspects of MEMS-based storage with the following two contributions. We devise policies to reduce the energy consumption of MEMS-based storage devices. We also propose to exploit knowledge of the expected workload in configuring the data layout of a MEMS-based storage device in order to increase the effective capacity. Both contributions target at satisfying the increasing demand for green and inexpensive storage devices.

In addition to the energy and cost aspects, we make sure that the response time and the lifetime of MEMS-based storage devices are competitive. The data-layout and energy-saving policies account for the timing performance by looking at configurations that do not compromise on the response time. With respect to lifetime, we devise probe wear-leveling policies that increase the lifetime of MEMS-based storage devices with minimal influence on the energy consumption and the response time.

The dissertation incorporates the conclusions from the study of the policies, and investigates the employment of MEMS-based storage devices in important types of mobile application. For predominately streaming applications, we also investigate the influence of buffering on the energy consumption, response time, and capacity of MEMS-based storage devices. We put the technology into perspective by comparing it to Flash memory and Hybrid (Disk–Flash) storage.

Our system-level research in this dissertation identifies potential points of enhancement of MEMS-based storage devices. Enhancements are targeted at reducing the energy consumption, decreasing the response time, cutting down the per-bit cost, and increasing the lifetime of the device. Most importantly, we show that the per-bit cost of MEMS-based storage is crucial to its success. Our system-level contributes to reduce the cost, while reduction on the device level is still needed.

We provide methods and means to configure MEMS-based storage devices to prepare them to serve in different environments as a viable storage technology. Following our research findings, designers can craft storage systems based on MEMS-based storage that are reliable, energy and performance efficient, and cost effective.

SAMENVATTING

Het digitale tijdperk waarin we leven eist een toenemende bewustwording van ons energieverbruik om de negatieve effecten daarvan op onze dierbare planeet te reduceren. Wij mensen organiseren ons doen en laten steeds meer met digitale middelen, wat de benodigde opslagcapaciteit meer dan ooit vergroot.

De bewustwording van milieueffecten en de wereldwijde dataexplosie vragen om een groene opslagtechnologie met extreem hoge datadichtheden. Een veelbelovende opslagtechnologie met deze eigenschappen maakt gebruik van MEMS (Micro-Electro-Mechanical Systems). Opslagdichtheden tot 4 Tb/in^2 zijn al aangetoond in het laboratorium. Met deze technologie wordt het mogelijk een opslagapparaat te maken ter grootte van een vingernagel met een capaciteit van 1 Tb en met een vermogensdissipatie van slechts één Watt.

Radicaal vernieuwende technologie vergt echter een behoorlijke tijd en inspanning voordat ze geschikt is voor toepassing in consumentenproducten. Een belangrijke oorzaak van deze vertraging is de moeizame integratie en acceptatie van nieuwe technologie. Breedgedragen integratie omvat onder meer onderzoek naar de plaats die de nieuwe technologie tussen bestaande technologieën kan innemen en het vinden van oplossingen voor de problemen die daarbij onvermijdelijk optreden. Vroege oplossingen voor het integratieprobleem bekorten de benodigde tijd tot de introductie op de markt en dragen hoogstwaarschijnlijk bij aan het algemeen succes van de nieuwe technologie.

Ter illustratie, Flash werd voor het eerst getoond in de jaren tachtig van de vorige eeuw. De enorme vraag naar Flash voor mobiele systemen stimuleert nu pas een enorme hoeveelheid aan onderzoek naar Flash. Zo wordt er pas recentelijk constructief onderzocht hoe men op basis van Flash-technologie betrouwbare, snelle, en energiezuinige opslagsystemen kan ontwerpen. Flash wordt nog weinig toegepast in de huidige generatie productiesystemen voor opslag, terwijl de alom toegepaste concurrent, de harde schijf, ondertussen onnodig veel energie verbruikt. Al met al kost de late reactie op de integratieproblemen met Flash dagelijks miljoenen euro's aan energie en koeling in datacentra.

Ter voorkoming van dergelijke late reacties voor MEMS-opslagapparatuur onderzoeken wij proactief hoe MEMS-apparatuur zo vroeg mogelijk kan worden geïntegreerd. MEMS-opslagapparatuur heeft, net als elke nieuwe technologie, haar eigen uitdagingen en optimalisatieproblemen. In ons onderzoek optimaliseren we het MEMS-opslagsysteem voor toepassing in mobiele, batterij-gevoede, systemen. Vervolgens vergelijken we onze resultaten met Flash-geheugen en met een hybride opslagsysteem, waarin we Flash combineren met een harde schijf.

Deze dissertatie concentreert zich op de energie- en kostenaspecten van MEMS-opslagsystemen. Onze belangrijkste bijdrages zijn methodes om het energieverbuik van dergelijke systemen te beperken. Voorts maken we bij het configureren van MEMS-opslagsystemen handig gebruik van voorkennis van de verwachte belasting, opdat de beschikbare capaciteit efficiënt wordt benut. Onze resultaten dragen bij aan de toenemende vraag naar groene opslagsystemen met extreem hoge datadichtheid. Bovendien beperken we de kosten, hetgeen acceptatie in de markt ten goede komt.

Naast de energie- en kostenaspecten zorgen we ervoor dat de reactietijd en de levensduur van MEMS-opslagsystemen kan concurreren met die van andere systemen. Onze data-layout en energiebesparingsmethodes hebben een gunstige invloed op de snelheidsprestaties; wij kijken expliciet naar configuraties die geen concessies doen aan de reactietijd. Wat betreft de levensduur van MEMS-opslagsystemen ontwikkelen we een methode voor wear-leveling[1] die de levensduur verlengt en tegelijkertijd het energieverbruik en de reactietijd slechts minimaal toe laat nemen.

Deze dissertatie bouwt voort op de conclusies van het onderzoek naar operationele methodes van MEMS-opslagsystemen en ze onderzoekt de praktische toepassing ervan voor enkele belangrijke mobiele toepassingen. Voor toepassingen met voornamelijk streaming data, zoals een videocamera, onderzoeken we onder andere de invloed van bufferen op het energieverbruik, de reactietijd, en de capaciteit van MEMS-opslagsystemen. We plaatsen de praktische betekenis van MEMS-opslagsystemen in perspectief door ze te vergelijken met Flash-geheugen en het eerder genoemde hybride opslagsysteem.

Ons onderzoek levert een aantal praktische aanbevelingen op voor MEMS-opslagsystemen. De aanbevelingen in deze dissertatie beogen een vermindering van het energieverbruik, een verkorting van de reactietijd, een verlaging van de kosten per bit en een verlenging van de levensduur van het systeem. Bovenal laten we zien dat de kosten per bit de cruciale factor zijn voor het succes van MEMS-opslagsystemen. Ons onderzoek draagt bij aan de kostenreductie op systeemniveau; dit laat onverlet een kostenreductie op componentniveau.

Deze dissertatie presenteert methoden en middelen voor de configuratie van MEMS-opslagsystemen, waarmee deze een levensvatbaar alternatief vormen als opslagsysteem in verschillende omgevingen. Met onze onderzoeksresultaten kunnen ontwikkelaars praktische MEMS-opslagsystemen bouwen die én betrouwbaar, én energiezuinig, én snel, én betaalbaar zijn.

[1] slijtagevereffening

ACKNOWLEDGEMENTS

Well, that is the end of an establishment period of my career, and I am still alive! Through this period I have experienced a lot on many accounts. Luckily, I was not alone during this period, but there were so many fantastic people that I worked with and met from all around the world. Had not it been for pursuing the PhD degree at Twente, I would not have got the opportunity to meet them.

The first person to meet is the person who accepted me to pursue PhD under his supervision, Pieter. You believed in me from the very first moment, and offered me the opportunity to carry out research in one of my beloved topics in Computer Science. Pieter, the unique of you is that you are an open person, easy going, and most importantly, that you let your students rule their own to prove themselves, while providing them with everything they need. I learned a lot from you, particularly the way to organize thoughts and to do research.

Hylke, you are, as I always describe you, an open, cheerful, honest, and faithful colleague. I always liked working and exchanging ideas with you. I absolutely learned from you, particularly how to frame my work for easy and effective communication, my innocent reader! I owe you getting the material of this thesis presented as they are now. I am also very privileged to have the chance to share office with you; I really had a nice time discussing with you technical and non-technical topics.

Gerard, thanks for your involvement in my work and for your comments on my thesis draft, which were very helpful. I also thank you a lot for teaming me up with the CAES group. I have had inspiring colleagues there.

I would like to thank Leon from Electrical Engineering, who gave the effort and time to review the thesis thoroughly. Your comments have improved its readability for people outside the field of computer science.

Berend-Jan, you are a nice colleague with whom I spend three years of my PhD. I benefited a lot from your review of my early papers. Frank, you have fantastic tweaking hands that helped me setting up the networking of the experimental setup. A lot of credit goes to you getting the network running properly. Johan, although we are two years out of phase, we still managed to get our PhD project moving properly with tangible results. Getting the modeling

in this thesis right would have been a tough endeavor without you, thanks a lot. Pascal, thank you for providing me with a LaTeX ground to build on and get the thesis in this nice outlook, I think.

I thank Haris Pozidis and Evangelos Eleftheriou from IBM Research Laboratories in Zurich for providing me with insights into MEMS-based storage, and answering my questions that helped shaping up the work. Also, I thank Charlotte Bolliger for her efforts to get me a high-quality picture to put on the cover of this thesis.

Steven Luitjens provided me with the measurement setup, which was an intrinsic part to carry out the experiments that are at the heart of this thesis. Thank you a lot for providing that and for hosting me for two weeks in your laboratory to get me into the setup.

During the course of my PhD I visited the University of California at Santa Cruz in the USA. Ethan accepted to work together on a research idea on MEMS and hosted me in the SSRC laboratory for a few months. That was undoubtedly a turning point for me. Ethan, your door was always open, and you were very supportive, I owe you a lot. I also wish to thank Darrell for being open and giving me from his time to answer some of my curious questions.

I would like to thank the user-committee members of my PhD project for their fruitful and useful feedback on my work through the project meetings we had.

Also, the comments I received from my graduation committee were very helpful to yet improve the quality of the thesis, thank you very much.

Many brilliant and mellow colleagues and friends were around me. Ayse, Delia, Ileana, Kavitha, Nirvana, Ozlem, Raluca, Vasughi, Albert, Jan, Michel, Mihai, Mohsen, Pierre, Philip, Supriyo, Trajce, and Qiang, thank you very much for the nice time and discussions we had together. Also the colleagues in DIES, CAES, PS, and TST-SMI groups were always there to give at least their supporting smile.

My colleagues and friends in Santa Cruz were so impressive and helpful. We could in a few weeks hang out, chat, and laugh. Andrew, Kevin, Mark, Rosie, Sasha, and Stefani thanks for being so nice colleagues and for getting me into playing Wii ;-).

I do not forget my brothers and sisters in the UT-Moslem community. Ziad, Mohammed Alhaj Ali, Mohammed Jaddallah, Ahamd, Wael, Ashraf, Abdulsalam, Dlovan, Rabah, Sameh, Muhammad Asif, Khurshid, Wissam, Muhammad Kamel, and many other. Thank you all for being nice friends. I thank Hajj Omar Elamad and his wife, Hajja Basma Alshourafa, for their hospitality and for their care for the community. I thank Hadi Tabaa for showing me how adding fun to everything turns it marvelous.

I thank my friends across the boarders in Germany who kept in touch with me to preserve the minimum of the nice time we had during our Master's program, Jihad, Mustafa, Yaser Daboul, Yaser Houri, Amjad, Lars, Bachar, Nizar, Meraa, Marwan, Talal, Younes, Momtez, Khaled, Obaida, and many others. I

also thank Wael Adi for his support whenever we spoke to each other. I would like to thank Said Krayem and Safa for their nice supporting emails.

Although during the course of my stay in the USA I was spending most of the time in Santa Cruz, I managed to climb the hill to meet amazing and inspiring personalities. In a very short time, we were talking the same language, laughing, and enjoying our company. The first and the foremost is Dian. You are just an amazing women and a productive sister that I am proud of and take honor in considering you as my elder sister. I am wordless to thank you and thank your husband, Ashraf, for your care and amazing devotion. Another fantastic couple is Houda and Isa. Wow, you guys are amazing, cheerful, and outgoing. I do not forget our discussions, so fruitful. I spend most of the time when I was in San José, with my friend and brother, Danny. Every time we hang out, he was prepared to pose on me some questions that required my explanations. Sometimes I filled the gap when no questions existed. Till the moment, I wonder whether you liked one of my eloquent explanations :-).

Gratitudes go to our fantastic secretaries, Marlous, Nicole, Nienke, and Thlema. Thanks a lot for helping and for bringing a nice atmosphere in the group(s). Nienke, you revolutionized the DIES group with your help, smile, and lovely surprises. The “Appeltaart” you make is unmatched!

My family in Syria was supporting me throughout every moment of the PhD. Every year I went Syria to visit them they received me with a big and warm smile, while noticing there was a laptop in my suitcase to work on even during vacation. They kept on supporting and stuffing me with all kind of delicious food. My mother was preparing a nutritious program dedicated for me. I was taking a new cooking every day to compensate for the loss in weight and depravity of nice food, for which I was lacking experience to prepare. My father was always checking how I am doing through the PhD course and kept encouraging me until the last moment. My brothers Basem and Safouan would always snap the chance of my presence in Syria to find out more about what I am doing for PhD and to pose very interesting questions to satisfy their curiosity. Laila, my only beloved sister, she also took a considerable part into my mother’s program to improve my conditions :-). I also do not forget your funny emails that brought my soul back to Syria for some moments to remember the nice times. You all are a part of this achievement. Had not it be for your support, I would not have managed through the past few years.

To the memory of my grandmother, Aliya, I love you! You left us four years ago, but you are still living in my heart. You were always supporting me and proud of me. I never doubted that I am your seventh son.

Thank you all,
Mohammed G. Khatib

27 April 2009, Enschede, The Netherlands

TABLE OF CONTENTS

Table of Contents **ix**

1 Introduction **1**
- 1.1 Green and High-Density Storage . . . 1
  - 1.1.1 Current Storage Technologies . . . 2
  - 1.1.2 MEMS-Based Storage Technology . . . 3
  - 1.1.3 Requirements for MEMS-Based Storage . . . 4
- 1.2 Research Statement: The Early Integration . . . 5
- 1.3 The A4 Project . . . 6
- 1.4 Contributions . . . 6
- 1.5 Organization of the Thesis . . . 9

2 Background and Related Work **11**
- 2.1 MEMS-Based Storage . . . 11
  - 2.1.1 Enabling Technologies . . . 11
  - 2.1.2 Designs . . . 13
  - 2.1.3 Architecture . . . 16
  - 2.1.4 Characteristics . . . 17
- 2.2 Complementary Storage Technologies . . . 19
  - 2.2.1 Flash Memory . . . 19
  - 2.2.2 Phase-Change Memory . . . 22
- 2.3 Related Work . . . 22
  - 2.3.1 Modeling . . . 23
  - 2.3.2 Design . . . 24
  - 2.3.3 Enhancement . . . 25
  - 2.3.4 Deployment . . . 27
- 2.4 Summary . . . 28

3 Research Methodology **29**
- 3.1 Experiment setup . . . 29
  - 3.1.1 The Experimental Platform . . . 31
  - 3.1.2 The Measurement Setup . . . 32
- 3.2 Modeling MEMS-Based Storage . . . 33

3.2.1 Model Parameters 35
3.2.2 Timing Performance Model 35
3.2.3 Energy Model 36
3.2.4 Data Layout 39
3.3 Workloads 43
3.3.1 Tracing Methodology 43
3.3.2 Traces 43
3.4 Summary 45

4 Enhancement Policies 47
4.1 Operation Modes 47
4.1.1 Contrasting with Hard Disk Drives 48
4.1.2 Power State Machine 50
4.1.3 Organization 51
4.2 Power Management Policy 52
4.2.1 Fixed-Timeout Power Management 52
4.2.2 Experimental Methodology 53
4.2.3 Simulation Results 54
4.3 Shutdown Policy 57
4.3.1 Two Shutdown Policies 58
4.3.2 Analytical Study 59
4.3.3 Trace-Based Study 62
4.4 Data-Layout Policy 63
4.4.1 Three Data-Layout Parameters 64
4.4.2 The Physical Subsector 64
4.4.3 Influence of Each Parameter 66
4.4.4 Design Space 72
4.5 Summary 74

5 Wear-Leveling Policies 75
5.1 Wear in MEMS-Based Storage 75
5.1.1 Types and Causes of Wear 75
5.1.2 Uneven Wear and its Effects 77
5.1.3 Device Life 78
5.1.4 Uneven Wear and Parallelism 80
5.2 Causes of Uneven Wear 81
5.3 Methodology 83
5.3.1 MEMS Translation Layer 84
5.3.2 Standard Deviation 85
5.4 The Sector-Based Round-Robin Policy 86
5.5 The Coldest-Probe Policy 89
5.6 The Barrier-Based Policy 92
5.7 Comparison 95
5.7.1 Device Lifetime 95

5.7.2 Timing Performance and Energy-Efficiency 96
5.7.3 Storage Requirement 98
5.7.4 Wear Leveling and Parallelism 98
5.7.5 MEMS Design Trade-offs 100
5.8 Wear-Leveling in Flash Memory 102
5.9 Summary 104

6 Capacity-Modest Applications 105
6.1 Methodology 105
6.2 Mixed-Media Environments 106
6.2.1 Configurations of the Data Layout 106
6.2.2 Results 108
6.3 Streaming Environments 109
6.3.1 Configurations of the Data Layout 109
6.3.2 Results 111
6.4 Enhancing MEMS-Based Storage Devices 113
6.5 Configuring for Streaming Applications 114
6.5.1 Bimodality of Request Size 115
6.5.2 Aggressive Prefetching 116
6.5.3 Exploiting the Bimodality 118
6.6 Summary 120

7 Capacity-Demanding Applications 121
7.1 The DISBA Architecture 121
7.2 The HYBBA and MEMBA Architectures 122
7.3 Traffic Shaping 124
7.4 Methodology 125
7.5 Comparison 127
7.5.1 Buffer Capacities 127
7.5.2 Energy Consumption 127
7.6 Energy-Efficiency of MEMBA 129
7.6.1 Streaming Demand 129
7.6.2 Best-Effort Demand 131
7.7 Reliability Study 132
7.7.1 HYBBA 132
7.7.2 MEMBA 134
7.8 Summary 135

8 Conclusions and Future Work 137
8.1 System-Level Conclusions 139
8.2 Architecture-Level Conclusions 140
8.3 Device-Level Recommendations 143
8.4 Future Work 145

List of Acronyms 147

Refereed Publications by the Author 149

List of Publications 151

A Excerpts from Collected Traces 159
A.1 I/O Trace 159
A.2 Power Trace 159
A.3 Control Log 160
A.4 Statistics of the I/O Traces 161

B Modeling Two Shutdown Policies 167
B.1 Modeling 167
B.2 The Energy-Efficient Policy 168
B.2.1 Motion along X 169
B.2.2 Motion along Y 171
B.3 The Performance-Efficient Policy 172
B.3.1 Shutdown time and energy 173
B.4 Implementation in DiskSim 174

C Modeling Streaming Architectures 175
C.1 Energy Components 175
C.2 Minimizing Energy 176
C.3 Buffer Capacities 178
C.3.1 The Real-Time Buffer Capacity 178
C.3.2 The Primary Buffer Capacity 180
C.3.3 The Secondary Buffer Capacity 181
C.4 Energy Consumption 182
C.4.1 Disk Energy Consumption 182
C.4.2 Flash Energy Consumption 182
C.4.3 DRAM Energy Consumption 183
C.5 Servicing Best-Effort Requests 184
C.5.1 Design Issues 184
C.5.2 Modeling the Best-Effort Demand 185

CHAPTER 1

INTRODUCTION

Man has recorded data for a very long time. Our ancestors carved their messages on tablets made of stone and wood. Stone and wood are durable media! It is no wonder why inscriptions carved several millennia ago are still readable today — but are not necessarily interpretable.

The digital era revolutionized the way how data are recorded. Several materials have been discovered that allow to record a huge amount of data on a small area. Yet, physicists are still pushing the envelope to achieve higher storage capacities with existing techniques as well as to innovate new storage technologies; the recently discovered Memristor [1] is just an example.

In the digital era, the amount of data a person stores has multiplied significantly. According to research companies, such as IDC[1], the overall demand for storage capacity is growing annually by about 30% – 60%. People keep track of their personal data for every event in their life. Further, laws, such as the US Sarbanes-Oxley Act of 2002 (SOX) [2] and the EU data retention directive [3], stipulate that an organization must retain its records for long periods of time.

1.1 Green and High-Density Storage

The digital era coincides with an era of increasing concerns about energy conservation. Our source of energy, namely fossil fuel, is limited. Sustainable energy sources are good alternatives, but they should be used wisely.

[1] `http://www.idc.com`

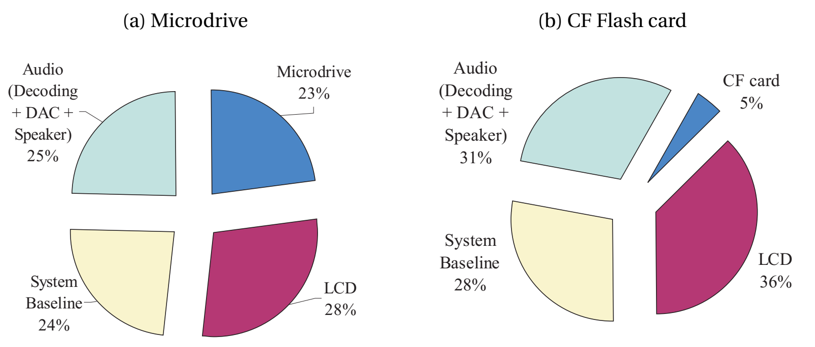


Figure 1.1: Energy breakdown of an HP iPAQ PDA measured in our laboratory when streaming at 192 Kbps from a Microdrive and a Flash card

Energy efficiency is an important design issue in a mobile computer system, since a battery has a limited capacity. Energy efficiency extends the time period during which a system is operational. Likewise, in data centers, energy efficiency reduces the direct costs, such as the energy consumption by a computer system itself, as well as indirect costs, such as cooling [4]. This is beneficial for economical and scaling reasons, since the capacity of data centers is mainly limited by their power demands [5].

The two most widely used non-volatile storage technologies to date are the Disk drive and Flash memory. A large proportion of the energy of contemporary computer systems is due to energy consumption of the inexpensive Disk drive. The more energy-efficient Flash memory is up to ten times more expensive than the Disk drive. The call for energy conservation and the demand for large storage capacities require *inexpensive green* storage systems.

1.1.1 Current Storage Technologies

The Disk drive has been around for more than 50 years serving as secondary storage in computer systems. The Disk drive is inexpensive. The initial price (or cost) of a Disk drive is 0.30 – 0.80 $/GB. Because of its mechanical nature, the Disk drive consumes a large amount of energy. Figure 1.1a shows that the smallest disk drive, the Microdrive, consumes about a quarter of the total energy consumed by a mobile computer system. The Disk drive has high endurance and can be written approximately 10^{12} times.

It is predicted that the areal density of the Disk drive will continue to grow by about 40% every year, whereas the linear density (i.e., track and bit density) grows by about 15%. As a result, the bandwidth is likely to improve also by 15%, but the access time (i.e., the latency) will hardly see any improvement (about 5% per year if any). The cost per gigabyte is expected to drop at a rate of approximately 40% per year. With respect to energy consumption, the Disk drive is expected to make little improvement [6].

Although it has already been introduced in the early eighties, Flash memory has only gained popularity in the past few years as its cost has dropped significantly. Flash memory is still more expensive than the Disk drive and costs 2.5 – 10.0 $/GB. Flash memory is, however, more energy efficient than the Disk drive, since it has no mechanical parts. Figure 1.1b shows that a Flash card consumes about 5% of the total energy consumed by a computer system. Flash provides orders of magnitude shorter access time than the Disk drive. A Flash cell can be written $10^4 - 10^6$ times, which is much less often than a disk.

Storage density of Flash is expected to grow as the technology node decreases, following Moore's law, and the number of bits per cell increases. Flash memory faces some challenges. Flash price should further decrease in order to provide competitive storage systems. The cost of a Flash-based storage system depends largely on the performance requirement, which determines the controller quality, the number of Flash chips, and the amount of buffering required. Flash endurance drops significantly when the Flash cell shrinks. Designers envisage that Flash memory will face significant challenges at technology nodes below 30 nm [7, 8], which has consequences for the endurance and retention of Flash[2].

1.1.2 MEMS-Based Storage Technology

MEMS-based storage is a new technology that has been proposed in the late nineties. MEMS-based storage promises inexpensive green high-density storage systems. The technology has emerged from Scanning Probe Microscopy (SPM), where a surface is scanned and manipulated at the atomic level. This storage technology is based on Micro-Electro-Mechanical Systems (MEMS). A MEMS-based storage device consists of two separate layers facing each other as shown in Figure 1.2. The moving layer carries the storage medium, which records data. MEMS-based storage devices employ high-density recording techniques that can achieve bit dimensions down to $1.5 \times 1.7\,\text{nm}^2$ [9]. The stationary layer is an array of thousands of read/write probes (or heads). IBM has prototyped a MEMS-based storage device with 4096 probes on an area of $41\,\text{mm}^2$. We describe MEMS-based storage in more detail in Section 2.1.

With respect to cost, MEMS-based storage devices have potentially low cost for three primary reasons. Firstly, they can be manufactured using the well established batch MEMS fabrication techniques [10]. Secondly, MEMS-based storage devices can be manufactured using micron-scale fabrication plants, whose equipment were installed ten years ago. The equipment of these

[2]Jan van Houdt from IMEC discussed Flash challenges in his presentation at the IMST (Innovative Mass Storage Technologies) event in November 2008.

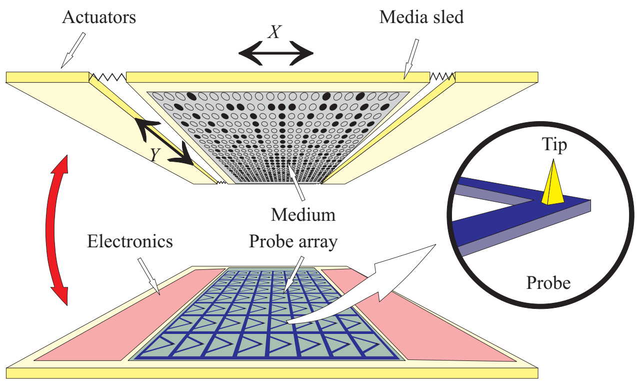


Figure 1.2: A three-dimensional sketch of a MEMS-based storage device

plants have passed their break-even point [11], avoiding the need to build dedicated fabrication plants, unlike for Flash. Thirdly, these plants can be used to produce future generations of MEMS-based storage devices, since MEMS poses no requirements on the lithography when increasing the density [11].

With respect to energy consumption, MEMS-based storage devices have a mechanical nature and therefore consume more energy than Flash and have inferior response time. Nonetheless, exploiting its unique characteristics, we make the case in this dissertation that MEMS-based storage can be used/configured to be competitive with Flash memory.

1.1.3 Requirements for MEMS-Based Storage

It is generally expected that the per-bit cost of storage will drive the memory market for decades to come [6], and thus the success of a new storage technology depends mainly on its per-bit cost. Additionally, a new storage technology should at least be equivalent to existing technologies with respect to timing performance, and energy-efficiency.

For example, (NAND) Flash memory replaced the Microdrive in handheld gadgets as soon as the cost of Flash dropped below that of the Microdrive. Flash could not make it before the price dropped, despite the fact that Flash has shorter response time, consumes less energy, and has better shock resistance. Another example is the Flash-based Solid-State Disk (SSD), which is not yet widely used in mobile computers. This is because the SSD is more expensive than the Hard Disk Drive (HDD), even through the SSD outperforms the HDD on other accounts.

We believe that MEMS-based storage can find its way to the deployment stage, *if and only if* it offers lower per-bit cost than Flash memory, and is, at least, as performance- and energy-efficient. Other factors, such as long retention time, high endurance, are relevant too. This dissertation offers several optimizations that enhance the energy-efficiency, timing performance, and lifetime of MEMS-based storage devices to the level where serious deployment of MEMS-based storage could be realized. We offer a technique to increase the retained capacity of a MEMS-based storage device after formatting. Nonetheless, the per-bit cost of MEMS-based storage technology is mainly in the hand of the device designers. Innovative recording techniques are required that enable high storage densities by shrinking the cell size or storing multiple levels per cell.

1.2 Research Statement

Our investigation of MEMS-based storage technology starts from the following hypothesis:

Hypothesis

Given its properties, a MEMS-based storage device can deliver appropriate quality of service as a non-volatile secondary-storage device in mobile computer systems.

We validate this hypothesis in the *context of mobile computer systems* (see Chapter 1.3). The validation considers a range of aspects for meaningful results.

Investigating MEMS-based storage as secondary storage includes optimizing the technology, so that it serves its task while delivering the best quality of service. Integrating MEMS-based storage raises the question of how it compares to existing technologies. The fact that MEMS-based storage technology is currently under development provides us with an opportunity to influence the design of the architecture and its components. Our research questions are:

Research Questions

Q1: How can a MEMS-based storage device deliver appropriate quality of service: high energy-efficiency, high timing performance, large capacity, and long lifetime?

Q2: How do MEMS-based storage devices compare to current storage devices composed out of Flash memory and the Disk drive?

Q3: What are the components of a MEMS-based storage device that determine its quality of service?

1.3 The A4 Project

The work in this dissertation is carried out in the context of the A4 project: **Adaptive non-volatile storAge for Adaptive Applications**. The A4 project is a joint research effort between the Electrical Engineering Department and the Computer Science Department at the University of Twente.

The basic idea of the A4 project is that adaptivity of the storage device meets with adaptivity of the application (and vice versa) for energy efficiency. One example of how adaptivity can save energy is that a MEMS-based storage device switches off some of its read/write probes that are not used for accessing the data requested by the application.

The A4 project targets mobile environments, which have stringent constraints on energy consumption, form factor, and shock resistance. *The focus of the project is on the integration of MEMS-based storage devices in mobile computer systems.* The study encompasses two parts, one at the system level and the other at the device level.

At the system level, the A4 project investigates various types of policy that enhance four aspects of a MEMS-based storage device: energy efficiency, timing performance, effective capacity, and lifetime. At the device level, on the other hand, various actuation techniques are investigated to enhance (some of) the previous aspects. The two parts complement and feed results to each other as shown in this dissertation.

1.4 Contributions

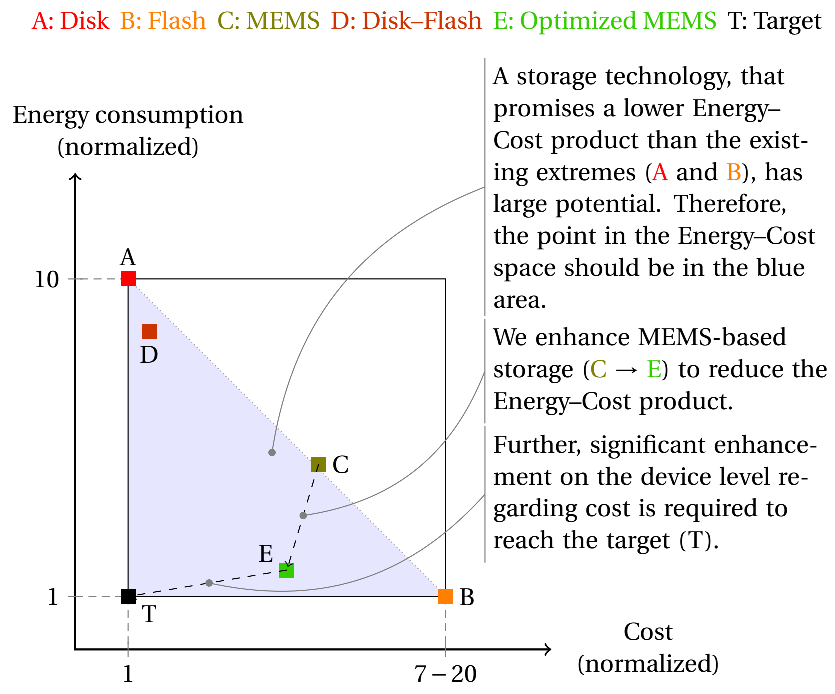
Figure 1.3 places existing storage technologies in an Energy–Cost space. The figure shows the Disk drive (point A) and Flash memory (point B) as the baseline from cost and energy perspectives, respectively. Figure 1.3 also presents the target technology (T) that offers inexpensive storage like the Disk drive and energy-efficient storage like Flash memory.

In a nutshell, our main contribution to MEMS-based storage is to reduce the energy consumption and reduce the per-bit cost by going from point C to point E in the Energy–Cost space. We also investigate the marriage between the Disk drive (A) and Flash memory (B) to construct Hybrid storage (D). We compare MEMS-based storage to Flash memory and to Hybrid storage.

Answering the three research questions, in the following, we summarize our contributions and mark the main ones with an asterisk:

Understanding We carefully study MEMS-based storage and investigate its unique characteristics. We survey the literature on the available designs of MEMS-based storage.

Analyzing The study of the characteristics results in identifying the operation modes of a MEMS-based storage device in the form of a Power State



A storage technology, that promises a lower Energy–Cost product than the existing extremes (A and B), has large potential. Therefore, the point in the Energy–Cost space should be in the blue area.

We enhance MEMS-based storage (C → E) to reduce the Energy–Cost product.

Further, significant enhancement on the device level regarding cost is required to reach the target (T).

Figure 1.3: A visualization of the relative position of each of the addressed storage technologies with respect to the per-bit cost and the per-bit access energy consumption. The Disk drive (A) and Flash memory (B) represent the baseline, respectively. The target storage technology (T) for mobile computer systems has the cost of the Disk drive, and is as energy-efficient as Flash memory. A non-optimized MEMS-based storage (C) is approximately seven times as expensive as the Disk drive, and sits approximately halfway in the range of the energy consumption. We devise enhancement policies for MEMS-based storage (E), so that it becomes competitive with Flash memory with respect to energy consumption. Further, we provide a technique to format MEMS-based storage, whereby most of its physical capacity is retained, so that it becomes less expensive. Nonetheless, device level enhancements are still needed to bring MEMS-based storage that last and crucial step toward the target (T). Current marriage between the Disk drive and the Flash memory (D) promises a greener and less expensive system than the Disk drive or the Flash memory alone. Note that the respective cost and energy consumption of the technologies changes from one environment to another (i.e., streaming and best effort), and from one storage system to another; and also changes over the time. This change reflects on the dimensions of the rectangle. We revisit the figure in Section 8.2 to reposition the technologies based on our results for capacity-modest and capacity-demanding applications.

Machine (PSM). The operation modes allow us to optimize a MEMS-based storage device systematically for good quality of service.

Modeling In a joint study with the device designers, we focus on the prototyped IBM MEMS device. We construct a model that approximates the IBM MEMS device, which we use throughout our simulation experiments. The model can be used by device researchers to test possible designs of various components of a MEMS-based storage device.

Optimizing* We optimize for four design targets of MEMS-based storage devices: energy consumption, response time, capacity, and lifetime. Our optimization transitions MEMS-based storage from point C to point E in the Energy–Cost space, see Figure 1.3. We offer four types of policy: a power management policy, a shutdown policy, a data-layout policy, and a wear-leveling policy. Each of the policies addresses one or more of the design targets as follows:

- **Energy** Power management transitions the device from the idle mode into the inactive mode, where the shutdown policy exploits the unique architecture of the device to shut it down in an energy-efficient manner. The data-layout policy stripes sectors across the probes, so that data are accessed efficiently.
- **Performance** We show that the response time of the device decreases if immediate shutdown decisions are avoided. We propose two shutdown policies that improve the performance for random and sequential workloads, respectively. The data-layout policy reduces the access time of requests by increasing probe utilization.
- **Capacity** Our optimization for the capacity is addressed by the data-layout policy, which strives at reducing storage overhead. We show that by exploiting knowledge of the expected workload the device can be formatted with large sectors, so that the overhead reduces and most of the physical capacity is retained.
- **Lifetime** We propose three light-weight wear-leveling policies that distribute write requests across probes, thereby avoiding the untimely demise of probes. These policies do not require complicated computations, and differ in the device lifetime they achieve versus the influence on the timing performance and energy-efficiency.

Configuring Investigating the devised policies under realistic workloads, we come up with design rules for each type of policy. The design rules assist the designers of MEMS-based storage devices in implementing informed configurations.

Integrating* Employing a MEMS-based storage device as secondary storage has a consequence for the amount of main memory required. For predominately streaming environments, we study the influence of aggressive prefetching on the configuration of MEMS-based storage devices which has consequences for the energy efficiency and the capacity.

Comparing* We study the response time and energy consumption of MEMS-based storage devices in two different environments: pure streaming, and mixed-media (i.e., a realistic mix of best effort and streaming applications). In both case studies, we compare MEMS-based storage with Flash memory. We also investigate a Disk–Flash Hybrid technology, and compare with MEMS-based storage.

Designing* Integrating MEMS-based storage devices in the storage hierarchy, we identify potential points of enhancement on the device level. We provide device researchers with these recommendations, so that research on the device level can be oriented purposefully.

1.5 Organization of the Thesis

Given the broad scope of the work (from application to device), Figure 1.4 offers a guide to reading each of our contributions separately. The content of the chapters can be summarized as follows.

Chapter 2 discusses the underlying technologies that enable MEMS-based storage. It also addresses complementary storage technologies, such as Flash memory. We detail the characteristics of MEMS-based storage devices, and detail our contributions to the existing system work on MEMS-based storage.

Chapter 3 presents the methodology adopted throughout the experiments of this work. We describe the setup, which we use to measure energy consumption of Flash memory, and to collect I/O traces. We present our MEMS model and discuss the characteristics of the traces we use for simulation. The modeling part of this chapter is joint work [Khatib: 7] with the Electrical Engineering Department of the University of Twente.

Chapter 4 defines the operation modes of a MEMS-based storage device. It presents three types of policy: power management, shutdown, and data layout. These policies are introduced to optimize for the energy consumption, timing performance, and capacity. This chapter subsumes four publications [Khatib: 2, 3, 5, 8]; the latter is joint work with the Storage Systems Research Center (SSRC) at the University of California at Santa Cruz in the USA.

Chapter 5 follows up on optimizing MEMS-based storage devices by tackling the challenge of uneven probe wear. It studies the influence of wear leveling on the timing performance and energy-efficiency. We propose three policies to extend the lifetime of MEMS-based storage devices. The policies have different trade-offs between lifetime, performance, and energy-efficiency.

Chapter 6 presents two case studies, where a MEMS-based storage device serves as secondary storage in capacity-modest mobile systems, such as PDAs. Simulation results of a MEMS-based storage device are compared with empirical measurements of a Flash card. We also present a quick and effective method to configure MEMS-based storage devices for streaming environments. The chapter is based in part on a publication [Khatib: 3].

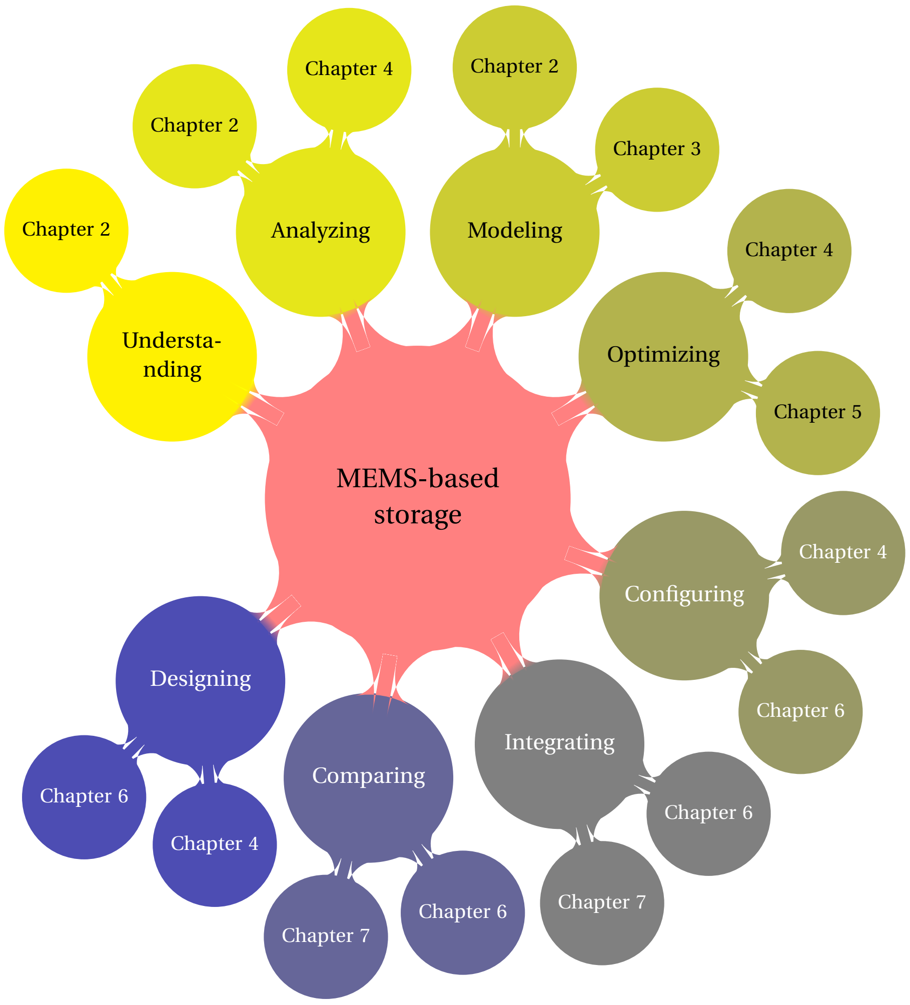


Figure 1.4: The contributions of this work to MEMS-based storage broken down into their respective chapters

Chapter 7 contributes to enhancing the energy efficiency of the storage hierarchy in capacity-demanding streaming systems, such as portable video players. We propose two approaches: (1) interposing Flash between the Disk drive and the DRAM, and (2) replacing the Disk drive with MEMS-based storage. We provide an analytical study that discusses the energy saving of the two approaches. This chapter subsumes two publications [Khatib: 1, 4].

Chapter 8 summarizes, gives a concise list of recommendations for the design on the device level, and concludes with a prospect of future work.

CHAPTER 2

Background and Related Work

This chapter introduces MEMS-based storage, and studies its characteristics. We present the proposed design models of MEMS-based storage and their common architecture. Two current competing storage technologies are also discussed along with their characteristics. Last, we discuss the work available on MEMS-based storage in the literature, and position our contributions.

2.1 MEMS-Based Storage

This dissertation investigates a recent storage technology, based on Micro-Electro-Mechanical Systems (MEMS), called MEMS-based storage. The technology is also called Probe-based storage particularly by electrical engineers; we use the term MEMS-based storage throughout the dissertation, since it is mainly used by system people, to whom this dissertation is mainly targeted. In fact, either term is a shorthand of "scanning–parallel-probe MEMS-based data storage", which states that we work on a storage technology that employs a large number of probes in parallel to access data from a storage medium. Components of the device are fabricated and integrated into one device using MEMS techniques.

2.1.1 Enabling Technologies

MEMS-based storage would not have been possible without the advent of two significant technologies. The first technology enables ultrahigh storage density, called probe recording. The second technology enables the realization of micro-machined storage systems. We explain them in the following sections. In addition to the two technologies, we present the proposed recording techniques.

Atomic Force Microscopy (AFM)

MEMS-based storage technology exploits a technique similar to Atomic Force Microscopy (AFM) to read and write bits. AFM deploys probes of nanometer-sharp tips to image, measure, and manipulate matters of a surface down to the atomic level. AFM was invented by Gerd Binnig, Calvin F. Quate and Christoph Gerber in 1986 [12]. In AFM, when the tip of a probe is brought into proximity of a surface, a force between the tip and the sample leads to a deflection of the cantilever of the probe. The deflection can be measured in several ways, such as using a laser spot reflected from the top of the cantilever into an array of photodiodes. Storing the measured deflections for a full scan of a surface results in a 3-dimensional profile of the surface. AFM has a high resolution in the order of fractions of a nanometer.

Contribution to MEMS-based storage The invention of AFM and its precursor Scanning Tunneling Microscopy (STM) [13] has opened a branch of microscopy, called Scanning Probe Microscopy (SPM). SPM forms an image of a surface using a physical probe to scan a specimen. SPM has inspired researchers to propose probe-based storage. The basic idea is that the ability to manipulate and sense matters of a surface enables the data storage. If after manipulation the change in matters during a certain time period is insignificant, non-volatile storage can be established.

Micro-Electro-Mechanical Systems (MEMS)

The second technology is Micro-Electro-Mechanical Systems (MEMS), which integrates mechanical elements, sensors, actuators, and electronics on a common silicon substrate. MEMS can be fabricated using modified semiconductor fabrication techniques. These techniques selectively etch away parts of a silicon (or other material) wafer and add layers to form the mechanical or electromechanical system. The research field of MEMS aspires to realize a complete computer system on a single chip [14]; researchers aspire to integrate the processor, the memory system, and a MEMS-based storage device into one chip in the future.

MEMS have an expanding range of applications [15]. Accelerometers and pressure sensors top these applications at present [16]. These sensors are deployed in automotive, medical, and consumer products. Radio-Frequency Micro-Electro-Mechanical Systems (RF-MEMS) are used to produce digital capacitors that can be tuned to different bands at run time [17]. Also, MEMS-based gyroscopes are used to park the head in the disk drive in the case of a sudden fall. Another application area is probe storage based on MEMS, which this dissertation contributes to.

Contribution to MEMS-based storage MEMS technology provides a miniaturized movement mechanism that is needed to scan and manipulate a surface. The storage medium is mounted on a sled, which is brought into position by micron-scale actuators [18]. As a result, the sled moves with respect to the probes, which read/write data. MEMS actuators differ in, among others, traveling speed and power dissipation. Therefore, the type of actuators used influences the timing performance and the energy consumption of a MEMS-based storage device.

Data Recording Techniques

In addition to the previous two technologies, that enable the read/write and actuation in MEMS-based storage, techniques for data recording are under investigation. Several recording techniques have been proposed for probe-based storage. Such recording techniques are magnetic-orientation switching [14, 19], topographic transformation [20], ferroelectric polarization [21], charge storage [22], and phase change [23].

2.1.2 Designs

Several design concepts for MEMS-based storage have been proposed and investigated. We discuss the Micro Scanning Probe Array Memory (μSPAM) concept proposed by researchers at the University of Twente, and the most advanced prototyped concept from IBM, namely the Millipede concept.

The μSPAM Concept

A large project has been established at the University of Twente to research MEMS-based storage: μSPAM [24]. The project represents a design concept of MEMS-based storage as envisioned by Twente researchers. The μSPAM project encompasses several groups: MEMS fabrication [25], probe development [26], tip-sample investigation [27], control [18], magnetic-medium development [28], and computer-system integration, which is the topic of this dissertation.

A μSPAM device has several storage media that move independently by electrostatic MEMS actuators, see Figure 2.1 for illustration. Each medium is accessible by a specific subset of the probe array. All storage media use a patterned magnetic technique, where one dot represents a bit. The patterned magnetic technique is under development at the Institute for Nanotechnology

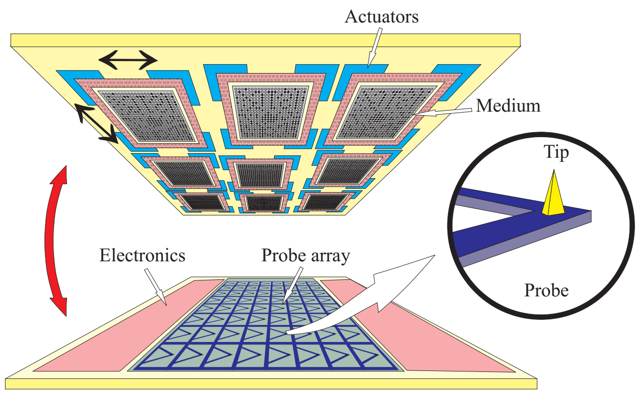


Figure 2.1: A three-dimensional conceptual view of the Micro Scanning Probe Array Memory (μSPAM) investigated at the University of Twente.

at the University of Twente (MESA+). The targeted main characteristics of the μSPAM device are presented in Table 2.1.

A μSPAM device employs magnetic probes. To write a bit, a probe comes into proximity of the magnetic medium. Due to the magnetic field of the probe, in combination with an external background field, the orientation of a magnetic dot is changed. The magnetization is out of the plane of the medium. The orientation of a dot represents the logical value of the bit. The read-out measures the magnetic forces created by the magnetic dot.

In ultrahigh densities, that exhibit a bit pitch in the order of 25 nm, addressing and accessing a track of bits at a high speed are challenging tasks. This is because the actuators must have a lateral resolution better than 25 nm in order to address densely packed bits with high precision. Another challenge is maintaining a constant distance between the tip and the medium in the order of 10 nm [27].

From a system perspective, the existence of several independently moving sleds gives a μSPAM device performance and energy advantages. Sleds can be moved to different locations to access data at the same time, reducing the access time. Parallel read and write requests can be dispatched simultaneously to different sleds for service. Further, unused sleds can be shut down (i.e., halted) to save energy.

The Millipede Concept

IBM has been working on MEMS-based storage for a decade in a project called Millipede. The Millipede project brought MEMS-based storage out of its infancy by realizing a prototype of a full MEMS-based storage device, which exhibits a storage density of 840 Gb/in^2 and a bit error rate of 10^{-4} [29]. IBM has

Table 2.1: Targeted characteristics of the μSPAM device and present characteristics of the prototype of the IBM MEMS device

Characteristic	μSPAM	IBM MEMS	Unit
Dimensions	$15 \times 15 \times 2$	$9.6 \times 9.6 \times 0.8$	mm^3
Total capacity	20	7.63	GB
Density	1000	840	Gb/in^2
Number of probes	128×128	64×64	probes
Per-probe data rate	10	40	Kb/s
Maximum data rate	20	20	MB/s
Seek time	< 2.5	< 1.5	ms
Read/Write power dissipation	< 1	≈ 1	W
Standby power dissipation	10^{-6}	0.005	W

recently demonstrated storage densities up to 4 Tb/in^2 [30].

Unlike a μSPAM device, an IBM MEMS device has one storage medium. The media sled is suspended across a probe array of 64×64 probes in the latest prototype. Bits are recorded in a thin polymer film as nanometer-scale indentations. Table 2.1 lists the key characteristics of a recent prototype.

An IBM MEMS device employs thermomechanical probes. An IBM MEMS writes a bit by applying a local electrostatic force on the tip of a probe and simultaneously heating the write resistor of the probe to 400 °C to soften the polymer locally. This process results in a nanometer-scale indentation, which represents a bit of value "1". During reading, a second resistor, mounted next to the tip, is heated in the range of 150 – 190 °C. When the tip moves into an indentation the thermal conductance between the cantilever and the media increases, resulting in a lower temperature of the heater, which can be detected by measuring its electrical resistance, and used as a read-out signal. To erase bits, thermomechanical effects relax the closely surrounding area of an indentation, so that it looses its depth and thus does not represent a "1".

Other Concepts

The promising ultrahigh density of probe-based storage has drawn the attention of academia as well as industry. In addition to the μSPAM and the Millipede concepts, several other concepts for MEMS-based storage have been proposed.

Carnegie Mellon University has proposed a single magnetic medium actuated by electrostatic actuators, called MEMStore. MEMStore has been intensively studied by system researchers and compared to disk drives. We discuss the work on MEMStore in Sections 2.3 and 3.2.4. Beside IBM, several indus-

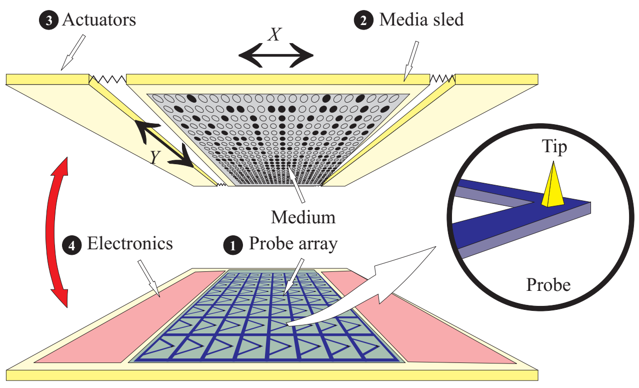


Figure 2.2: A three-dimensional conceptual view of a single-sled MEMS-based storage device

tries have been working on MEMS-based storage, including LG Electronics, Samsung, and Nanochip. LG Electronics proposed the Nano Data Storage System, that employs several arrays of probes for thermomechanical data storage on a polymer medium [31]. Samsung, jointly with Korean universities, is developing the Resistive Probe Storage System, that applies transistor technology to record bits on ferroelectric media [32]. Nanochip is developing a storage system to record bits on a phase-change medium. Nanochip is expecting the debut of its first product in 2010 [11].

2.1.3 Architecture

Several design models for MEMS-based storage have been proposed. Despite that these models employ different recording and actuation techniques, they have a common architecture. A MEMS-based storage device consists of two separate physical layers, one above the other, as shown in Figure 2.2. The top layer, the **media sled**, is suspended by springs across the bottom layer. The bottom layer is a two-dimensional array of read/write probes, the **probe array**. In addition to the storage medium and the probe array, a MEMS-based storage device has two other components, namely the **actuators** and **electronics**.

The probe array typically has several thousands of probes. The probe array is stationary and connected to the electronics. Increasing the number of parallel probes, enhances the throughput. The probe must have a nanometer-sharp tip to address a bit location on the medium with high precision. An imprecise addressing overwrites neighboring bits and thus damages data. Further, a probe tip should be durable to withstand wearing conditions for a sufficiently long lifetime.

Bits are recorded using different techniques. Such techniques are magnetic, phase change, and thermomechanical manipulation. The sled, which carries the storage medium, moves independently in the X, Y, and Z directions relative to the probe array. In all design models, each probe sweeps over a bounded area of the media sled, called the **probe (storage) field**. The movement along the Z direction is controlled per probe, unlike the other two directions. Consequently, seek times shorten and a relatively high (aggregate) data rate is attainable by operating many probes simultaneously.

The third main component of a MEMS-based storage device encompasses the micron-scale X, Y, and Z actuators. The media sled is attached to actuators to displace the storage medium relative to the probe array, so that data can be read from and written to the storage medium. Actuators should be able to operate at a high resolution in the sub-nanometer range to address nanometer-scale bits in ultrahigh-density storage. Since motion is necessary to access data, the actuators should allow for high access speeds at low energy consumption. Actuators can be, for example, electromagnetic as in the IBM MEMS device [29] or electrostatic [33].

The electronics is the fourth component, which is integrated with the other components. The electronics encompass the analog front-end, multiplexers, and digital components that error-correct data, control the mechanics, carry out housekeeping tasks, and interface the device to the host computer system. The interface accounts for the inactive energy, which constitutes a large fraction of the total energy, when power management is implemented.

2.1.4 Characteristics

A thorough understanding of the characteristics of a new technology, including its limitations and challenges, is of paramount importance for a successful employment of the technology. This is also the case for MEMS-based storage technology. We discuss its characteristics with respect to the medium, the probes, the actuators, and the device as a whole. We label these characteristics for ease of reference later.

MEMS-based storage devices have common characteristics, which stem mainly from their common architecture as well as their basic AFM and MEMS technologies detailed previously. In the following, we lay out these characteristics and point out their consequences for the timing performance, energy consumption, capacity, and lifetime of MEMS-based storage devices.

M1: Spring-suspended sled The media sled is suspended by springs across the probe array in MEMS-based storage devices (see Figure 2.2). Unlike disk drives, which spin up the medium before the head can be positioned across it, in MEMS-based storage devices the probe is always positioned directly across the medium. Therefore, MEMS-based storage devices have no mechanical startup overhead.

M2: Light-weight sled The media sled is the only moving part in a MEMS-based storage device. The mass of the media sled determines the time to accelerate and decelerate the sled. Consequently, it determines the amount of power, needed to generate sufficient force for actuation, and thus influences the energy consumption.

M3: Low per-probe data rate The data rate a probe can sustain is limited by several factors including the resonance frequency of the probe and the time to switch a bit on the medium between "0" and "1". Consequently, the per-probe data rate is inherently low; the per-probe data rate in the current prototype of the IBM MEMS device is 40 Kb/s [10]. A MEMS-based storage device employs an array of parallel probes to attain high aggregate data rates.

M4: Regularly spaced probes Probes are mounted on one single substrate at equal distances from each other. In the IBM MEMS device, probes are separated by 100 μm in the X and Y directions. This results in square probe fields of the same dimensions; for example a field is 100 μm × 100 μm in the IBM MEMS device. As a result, probe fields contain the same number of bits and thus have the same capacity.

M5: Large number of probes The large number of probes sharing one single medium limits the sweep area of a probe to a small area on the storage medium called the probe storage field. As a result, seek times become shorter and thus the seek energy decreases.

M6: Nanometer-sharp probe tips As detailed previously, AFM probes must have a sharp tip in the order of a nanometer for a precise scan, manipulation, or measurement of a surface. Such tips can pass an electrical current, make contact with the storage medium, and/or heat up, depending on the recording technique. As a result, a probe tip can loose its sharpness and/or its coating material, resulting in probe wear. As a probe is used more often, it wears faster and its lifetime becomes shorter.

M7: Separate X and Y actuations A MEMS-based storage device deploys two separate pairs of actuators to move the media sled independently in the X and Y directions at the same time. As a result, the time to move the sled from one position to another, known as the seek time, is the maximum time to seek along the X and Y directions.

M8: Z nanopositioners The spacing between the probe tip and the medium is dynamically maintained by nanopositioners. Because of the roughness of the medium, nanopositioners are used to maintain a nanometer-scale constant spacing, which is necessary for bit manipulation in contactless MEMS-based storage devices. Nanopositioners avoid a probe tip accidentally hitting the medium. In the IBM MEMS device, the probe makes contact with the medium only at read/write time using electrostatic force. Typically, the spacing between the tip and the medium in the IBM MEMS device is in the order of hundreds of nanometers [29], so that accidental hits are hardly possible.

M9: Mechanical nature A MEMS-based storage device has a moving media sled. To achieve high throughput, a MEMS-based storage device should move the sled as fast as possible, and should not interrupt the sled motion whenever possible. Therefore, data are laid out along one direction, conventionally taken as the Y direction in this work.

M10: Ultrahigh density Ultrahigh densities of over $4\,\mathrm{Tb/in}^2$ can be achieved with MEMS-based storage [29], resulting in large-capacity, small-form–factor devices employable almost everywhere. However, as the density increases, the time to position a probe above the center of a track of bits accurately, known as the **settling time**, also increases. A precise positioning is necessary for data integrity.

M11: Error proneness Storage devices are error prone for many reasons, such as errors in the medium and probes. Therefore, a MEMS-based storage device should add error-correction bits to user data and encode them before writing to the medium [29]. Error detection and correction occur at read time. The increase in reliability clearly comes at the cost of an increase in response time and energy consumption as well as a decrease in the effective capacity.

Having explained the main topic of this dissertation, namely MEMS-based storage device, we provide next a brief discussion of Flash memory and Phase-Change memory. MEMS-based storage has to be competitive with these two technologies.

2.2 Complementary Storage Technologies

Several storage technologies have been proposed. These technologies are at different stages in their development course spanning from fundamental investigation to advanced development, and end in deployment. Memristor memory [1] and racetrack memory [34] are in the early fundamental stage, whereas Phase-Change Memory [35] is still in the development stage. On the other hand, MEMS-based storage is nearing deployment and Flash memory is already deployed.

2.2.1 Flash Memory

Flash was invented by Fujio Masuoka while working for Toshiba in 1980 [36]. Invention of Flash was motivated by the view that the cost per bit is the key driving factor of the memory market. Flash memory uses the same principles of operation as the byte-addressable Electrically Erasable Programable Read-Only Memory (EEPROM). The main invention of Flash memory was,

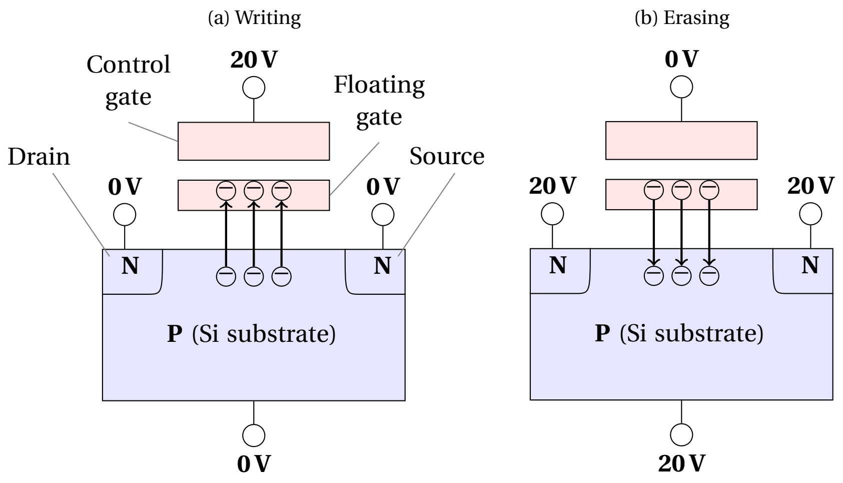


Figure 2.3: Writing (or programming) and erasing in Flash memory is achieved by tunneling between the floating gate and the substrate. Observe the different voltages applied at the terminals when writing and erasing.

to cut on the cell area and thus its cost, one of the two transistors in the byte-addressable EEPROM cell is removed. This transistor is responsible for the erasure on a bit level. Therefore, Flash erases on larger granularity than one bit, and, back then, was called the simultaneously erasable EEPROM. The name "Flash" was suggested later in 1984, since Flash has to erase a block simultaneously, which resembles the flash of a camera.

A Flash cell basically consists of a Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET) transistor. The transistor has two gates: the control gate and the floating gate. The floating gate is interposed between the control gate and the MOSFET channel, and electrically isolated from the source–drain channel by a layer of silicon dioxide. Storing data relies on trapping electrons in the floating gate. Writing a Flash cell turns a logical one into a zero by injecting electrons from the silicon substrate through the insulation layer into the floating gate as shown in Figure 2.3a. The insulation layer prevents leakage of electrons, hence the non-volatility property. Read-out applies a voltage to the control gate and senses the flow of the current from the source to the drain, which is controlled by the floating gate. Erasure applies a voltage of an opposite polarity to the write voltage to pull the electron off the floating gate as Figure 2.3b shows.

Two types of Flash memory exist: (1) NOR Flash, and (2) NAND Flash [36]. They differ in their exposed interface as a memory, and the internal structure of the interconnections between cells. In NOR Flash, cells are connected in parallel, resembling a NOR gate, whereas in NAND Flash, cells are connected in series, resembling the functionality of a NAND gate. The differences between the two types stem from the essential functionality each type is assumed to serve. While NOR Flash is developed to function as a Read-Only Memory (ROM) to replace EEPROM, NAND Flash is developed to function as a storage device to replace magnetic disk drives. Therefore, performance is the first design criterion for NOR Flash, whereas it is the per-bit cost for NAND Flash, since it is competing with the disk drive. In this work, which investigates a new class of non-volatile storage devices, we experiment with and compare to NAND Flash. Henceforth, we use NAND Flash and Flash interchangeably.

NAND Flash has no mechanical components, and, therefore, exhibits short latency, consumes low energy, and has high shock resistance. However, like any storage device (or memory), NAND Flash memory has its own characteristics, which stem from its fundamental principles as well as its structure. These are:

F1: Block-level erasure NAND Flash memory is organized internally in terms of blocks typically of size 16 KB to 512 KB. A block can only be erased as a whole, reducing the number of control transistors and interconnection in order to cut down the area and thus the cost. Erasing in Flash turns logical zeros back into ones.

F2: Page-level read/write Blocks in NAND Flash are further organized into pages. A block can contain 32, 64, or 128 pages, each is typically 512, 2048, or 4096 bytes in size. Pages in a block can be read or written individually. Writing in Flash turns logical ones into zeros.

F3: In-block sequential write Cells in NAND Flash are connected in series to reduce the necessary address lines and thus cut on cost. Because of the serial connection, pages in one block should be written in a sequential order to reduce disturb effects [37] that cause errors.

F4: Erase before rewrite Writing turns logical ones into zeros, which is done at the page level in NAND Flash. However, the reverse operation of turning zeros back into ones, called erasing, can only be done at the block level. The difference in size between the erasure granularity and the read/write granularity makes overwriting a single page in NAND Flash impossible. Flash provides a workaround solution by writing the updated page to a new block and migrating the other valid pages from the old block to the new one. This migration can be quite expensive in terms of energy and performance [38] as we will see in Section 6.3.2.

F5: Error proneness The read or write operation of a NAND Flash memory can spontaneously fail. An appropriate Error-Correction Code (ECC) is used to compensate for this failure.

F6: Endurance Erasing of a NAND Flash cell requires a high voltage to inject electrons back into the substrate. The high voltage causes a gradual degradation of the oxide insulating material [37], known as cell wear. As a result, the endurance of Flash cells decreases. Typically, a NAND Flash cell has a cycle rating of $10^5 - 10^6$ erasure cycles.

2.2.2 Phase-Change Memory

Phase-change memory uses materials that can be switched between two primary stable states: amorphous and crystalline. The transition between the two states is achieved controllably through prescribed annealing sequences. The material exhibits different properties depending on the state. It exhibits low resistivity and high reflectivity when crystalline, and high resistivity and low reflectivity when amorphous. The reflectivity property is exploited in optical storage devices, such as Compact-Disk Read-Only Memory (CD-ROM). Phase-change memory, on the other hand, exploits the resistivity property.

Phase-change memory promises higher endurance than Flash memory. Early experiments show that, like Flash memory cells, Phase-Change memory cells can store multiple levels, and can be stacked to reduce the per-bit cost. Phase-change memory has a faster switching time than flash, but dissipates more power. Recent discoveries of new materials, that allow for access times in the order of a nanosecond, has renewed the interest in Phase-Change memory. Several research laboratories, such as Numonyx, Intel, and IBM, are developing phase-change memory at present. To the best of our knowledge, these works improve the access time, power dissipation, and scalability of the Phase-Change memory cells [35].

It is, however, not yet clear how the cells will be connected to construct a complete memory system, which may result in a few types. This is exactly the case with Flash memory in its two forms NOR and NAND. Decidedly, the internal structure of a Phase-Change memory is a whole research topic in its own, so that it is far in the future to compare to. On the other hand, since Flash is already existing, we limit ourselves in this work to compare MEMS-based storage to (NAND) Flash.

2.3 Related Work

MEMS-based storage has been the subject of research in many institutions worldwide. Zhu [39] gives a survey of the state of the art in 2003. We can classify the work available in the literature on MEMS-based storage into four main categories: (1) modeling, (2) design, (3) enhancement, and (4) deployment of MEMS-based storage devices.

The modeling category constructs timing performance and energy models of MEMS-based storage devices, whereas the design category studies the influence of changes in the parameter settings on the performance- and energy-efficiency. The enhancement category investigates policies to enhance the performance and energy efficiency of MEMS-based storage devices. The deployment category investigates roles of MEMS-based storage devices to enhance the performance- and energy-efficiency of the whole computer system.

2.3.1 Modeling

MEMS-based storage device are not yet available for researchers to experiment with. Therefore, researchers have constructed models of MEMS-based storage devices to study their integration into computer systems. One of the advanced models, which was constructed jointly by system and device researchers, is the model of MEMStore, a MEMS-based storage concept developed at Carnegie Mellon University (CMU) [40]. The MEMStore model incorporates timing performance and energy models of MEMStore. It computes the timing performance and energy consumption of the seek, read, and write operations. The energy model further captures the energy consumed during idleness, inactivity, and startup. The MEMStore model has been integrated into the well-established DiskSim [41] storage simulator.

The majority of work on the MEMStore model comes from CMU. In addition, contributions from different institutions have advanced the model. Madhyastha *et al.* [42] compare three different seek models, one of which is the initial seek model of the MEMStore model. They propose a control model, that is based on closed-loop control and thus has a high precision in approximating the seek operation in a real MEMS-based storage device. Hong *et al.* [43] devise an analytical solution of the model, which has been incorporated in the MEMStore model. The model of the read and write operations are simpler than that of the seek model, since the media sled travels at a constant speed when reading or writing. Further, researchers have advanced the model by supplementing it with more features, such as data-layout policies, scheduling policies, which we address in the next sections.

Contribution Today, prototypes of MEMS-based storage do exist, particularly a prototype of the IBM MEMS device. Despite of their existence, these prototypes are unavailable for us. Therefore, we use trace-driven simulations to evaluate our contributions. For better accuracy, we refine the MEMStore model, update it to capture the state-of-the-art of MEMS-based storage, and set its parameter with figures from the IBM MEMS device. The reason of the choice of the IBM MEMS device is twofold: (1) it is the most advanced MEMS-based storage device today, and (2) enough details are available on the IBM MEMS device in the literature. Hence, our research does not become stale nor unrealistic. Table 2.2 presents the settings of the second generation of the CMU MEMStore and the IBM MEMS device [10]. The table shows a clear difference in the performance characteristics, particularly with respect to the per-probe data rate.

Table 2.2: The difference in characteristics between the second generation of the CMU MEMStore and the IBM MEMS device. G2 MEMStore is the most used design in the previous studies

Characteristic	G2 MEMStore	IBM MEMS	Unit
Sled acceleration	803.6	55.7	m/s^2
X settling time	0.2	0.2	ms
Per-probe data rate	700	40	Kb/s
Maximum data rate	560	20	MB/s
number of probes	80×80	64×64	probes
active probes	640	≤ 4096[a]	probes
Bit width	40	25	nm
number of sleds	1	1	sled
per-sled capacity	4.0	7.6	GB

[a] The number of active probes is a parameter we investigate. It depends on the packaging format, which has a certain power budget.

This work also contributes to the advancement of the MEMStore model, mainly to its energy part. Because the media sled in suspended by springs, the amount of power varies depending on how far the sled is from the center. We incorporate an energy model that takes into account the varying power across the medium, unlike the initial model, which assumes a constant power dissipation. Further, due to the spring structure MEMS-based storage devices lack startup overhead. MEMS-based storage devices, however, exhibit a shutdown overhead to bring the sled stationary. We refine the power-state machine of the MEMStore model by removing the startup state and adding a shutdown state. The initial MEMStore performance model assumes an instantaneous shutdown time. We contribute with two shutdown policies and incorporate their performance and energy models. Our contributions to the model are explained in Section 3.2.

2.3.2 Design

System researchers have carried out several sensitivity studies to help the designers of MEMS-based storage devices to focus on issues of importance to the overall performance of the device. Griffin *et al.* [40] studies the influence of the per-probe data rate, settling time, and spring force on the timing performance of a MEMS-based storage device. Likewise, Sivan–Zimet [44] studies the influence of the X and Y dimensions, and the number of active probes on the timing performance. Later, Dramaliev *et al.* [45] extend on Sivan-Zimet's study, and propose analytical models for fast exploration and evaluation of the influences. Griffin *et al.* study the parameter sensitivity statically, whereas Sivan-Zimet's and Dramaliev *et al.'s* study are dynamic, driven by traces.

Contribution Like the previous works, we also carry out sensitivity studies. Unlike these works, we study the influence of data layout parameters on the timing performance, the energy consumption, and the effective capacity. In addition to the number of active probes, we study the influence of the number of simultaneously accessible sectors, and the sector size. We study the influence under real-world traces.

Our studies for two types of workloads reveal the necessity of enhancement on the device level of a MEMS-based storage device. The system-level study results in a list of components of primary importance to enhance the performance- and energy-efficiency of MEMS-based storage devices. We provide this list in Section 6.4.

2.3.3 Enhancement

A body of work exists that enhances the performance- and energy-efficiency of MEMS-based storage devices. This work contributes with policies with respect to: (1) the logical data layout, and (2) I/O request scheduling.

Schlosser *et al.* [46] devise a data layout for MEMS-based storage devices (see Section 3.2.4), which we adopt throughout this work. They map logical addresses onto physical sectors such that mechanical overheads are reduced for sequential access patterns. Yu *et al.* [47, 48] leverage the parallel probes in MEMS-based storage devices to support performance-efficient accesses to relational databases. They show that, unlike disk drives, MEMS-based storage devices enable the access of two-dimensional data sets in both row-wise and column-wise fashions, elevating the I/O utilization as well as the cache performance.

With respect to scheduling, Griffin *et al.* [49] investigate the applicability of the scheduling policies from disk drives to MEMS-based storage devices. Such policies are First-Come, First-Served (FCFS) and Shortest Positioning Time First (SPTF). They show that these policies can be employed in MEMS-based storage devices. They observe the same respective performance as in disk drives; FCFS and SPTF perform the worst and the best, respectively. Schlosser *et al.* [50] propose Shortest Distance First (SDF) as a scheduling policy, which is specific to MEMS-based storage devices. This policy selects requests with the minimum Euclidean distance from the current position of the sled in both X and Y directions. Their experiments show that SDF performs worse than SPTF, and even other moderately performing policies. The reason is that SDF discards the fact that, like in disk drives, in MEMS-based storage devices re-positioning (along the X direction) incurs expensive settling. As a result, Schlosser *et al.* conclude that scheduling policies from disk drives, which optimize along one direction, work well for MEMS-based storage devices. Hong *et al.* [51] build on Sclosser *et al.'s* results and further customize SPTF for MEMS-based storage devices. They logically cluster the medium into several zones of different seek time profiles, and propose Zone-Based Shortest Positioning Time First (ZSPTF), which prioritizes requests to the currently visited zone over requests to other zones.

These works reduce the seek energy, since mechanical overheads are reduced. There are two works, which address the energy consumption of MEMS-based storage devices explicitly [52, 53]. Schlosser *et al.* [53] compare the energy consumption of MEMS-based storage devices and disk drives. Using file-system benchmarks, they show that MEMS-based storage devices consume $10\times - 50\times$ less energy than disk drives. To save on the idle energy, Lin *et al.* [52] propose a power state machine of one low-power mode, and evaluate the energy saving by varying the shutdown timeout in the range of 0 – 50 ms. They use simulation driven by traces from server workloads. Lin *et al.* recommend to shut down the device immediately — by setting the timeout to zero — after request completion for maximum energy saving at an increase of 0.5 ms in response time on average.

Contribution Unlike previous works, we do not optimize the seek time in MEMS-based storage devices. This is because we find that in all works on scheduling researchers scale up the request arrival rate up to 30 times, in order to be able to show the difference in the performance between their scheduling policy and other ones. Conservatively speaking, at a scaling factor of 5 FCFS and SPTF hardly show a difference in response time, whereas ZSPTF and SPTF show a difference starting at a scaling factor of 15. Researchers use traces from server workloads, which inherently exhibit high arrival rates. Despite of that, upscaling is still necessary. On this basis and since we target mobile environments, we employ the FCFS policy throughout this work.

We turn our focus to the read/write time, which is the time to read from or write to the storage medium. This is because, as will be shown in Section 6.2.2, the read/write time and energy predominate response time and energy consumption, respectively. In Chapter 4, we formulate the configuration problem of the data layout in MEMS-based storage devices using three parameters. And we propose to exploit knowledge of the expected workload to configure these parameters, leading to enhancement in performance, reduction in energy consumption, and increase in the effective (user) capacity.

To the best of our knowledge, this is the first work that addresses an advent challenge in MEMS-based storage devices. Probes in MEMS-based storage devices are susceptible to wear. A probe in MEMS-based storage devices accesses a storage field typically with a size of several tens of megabytes. Consequently, if a probe wears out, a whole field becomes inaccessible. As a further consequence, the performance of the device degrades and its energy consumption increases, since fewer probes are left to do the same amount of work as before. We address probe wear in Chapter 5 and introduce three wear-leveling policies of different influence on the performance, energy consumption, and lifetime of MEMS-based storage devices.

2.3.4 Deployment

Several roles have been proposed for MEMS-based storage devices: as (1) a disk cache [54], (2) a streaming buffer and cache [55], and (3) a replacement for disk drives [53], and (4) a full and partial replacement of disks and Non-Volatile Random Access Memory (NVRAM) in disk arrays [56].

Wang *et al.* [54] show that employing a MEMS-based storage device as a cache for disk drives with a capacity of just 3% of the disk capacity, the I/O subsystem has the performance of MEMS-based storage devices 30% to 49% of the time and exhibits the disk performance in the rest of the time. Rangaswami *et al.* [55] use MEMS-based storage devices as a buffer (and a cache) in streaming servers to reduce the demand for the expensive volatile Dynamic Random Access Memory (DRAM). As a result, the DRAM buffering requirement decreases significantly up to 90% depending on the number of simultaneous streams. Schlosser *et al.* [53] replace disk drives by MEMS-based storage devices. They show that I/O stall times reduce by a factor of 4 – 74 times over disks, and the overall application runtimes improve by a factor of 1.9 – 4.4 times over disks. Uysal *et al.* [56] offer an extensive study of various roles of MEMS-based storage devices in disk arrays for server application. By varying the MEMS-to-disk cost ratio in the range 1 – 10, they show that a full replacement of disks in a disk array improves the performance-to-cost ratio by a factor of 2 – 7. A partial replacement results in a factor 2.5 – 5.0, whereas a full replacement of the NVRAM has a factor of 2.1 – 4.2.

Contribution The small form factor, the low cost, and the high density of MEMS-based storage devices have motivated us to investigate the employment of these devices in battery-powered mobile systems. We distinguish between two types of applications based on capacity: (1) capacity-modest applications and (2) capacity-demanding applications. Capacity-modest applications are exampled by handheld devices, which required capacities in the order of tens of gigabytes today. On the other hand, capacity-demanding applications are exampled by portable video players, which demand capacities in the order of hundreds to thousands of gigabytes, today, to store high-definition digital contents. While Flash memory dominates the former type, the Disk drive dominates the latter due to cost reasons.

For capacity-modest applications, we prepare an experimental setup to capture I/O traces on a PDA to drive our simulations (Chapter 3). Despite of its high cost, NAND Flash memory is the dominant storage technology in mobile systems. Therefore, we compare MEMS-based storage to Flash. Using our setup, we measure the performance and energy consumption of a Flash memory for comparison. We demonstrate the competitiveness of a single-chip MEMS-based storage device with a Standard CompactFlash (CF) card with respect to performance and energy consumption in Chapter 6.

For capacity-demanding applications, we analytically investigate three different streaming architectures that are optimized for energy efficiency (Chapter 7). The first architecture we study is the conventional DISk-Based Architecture (DISBA). The energy saving of DISBA is, however, limited. We propose two approaches to the energy-saving limitation of DISBA, one evolutionary, and another revolutionary approach. The evolutionary approach combines the Disk drive with Flash memory in a HYBrid-storage–Based Architecture (HYBBA). The revolutionary approach replaces the Disk drive with future MEMS-based storage in a MEMs-storage–Based Architecture (MEMBA).

2.4 Summary

MEMS-based storage is an emerging storage technology that promises high-capacity small–form-factor storage solutions. MEMS-based storage technology relies mainly on two technologies: SPM and MEMS. Several design concepts of MEMS-based storage have been introduced by academia and industry. These concepts adopt different recording technologies. IBM has realized its design concept in a complete prototype with an 840 Gb/in^2 storage density. Sufficient material is available on the IBM MEMS device, which we take as a point of reference in our models and simulations.

A body of work exists on MEMS-based storage, to which we contribute. We show that MEMS-based storage devices have unique characteristics that distinguish them from other mechanical storage devices, such as HDDs. These characteristics allow to make MEMS-based storage devices performance- and energy-efficient. MEMS-based storage faces the challenge of uneven probe wear, for which we provide solutions in this dissertation. We investigate the deployment of MEMS-based storage devices in two different mobile application areas.

CHAPTER 3

RESEARCH METHODOLOGY

This chapter presents the experimental setup we casted to evaluate MEMS-based storage devices in capacity-modest applications. It details the timing performance and energy models of MEMS-based storage and the settings we apply. The last part discusses the tracing methodology and three traces we use throughout this dissertation for simulating a MEMS-based storage device.

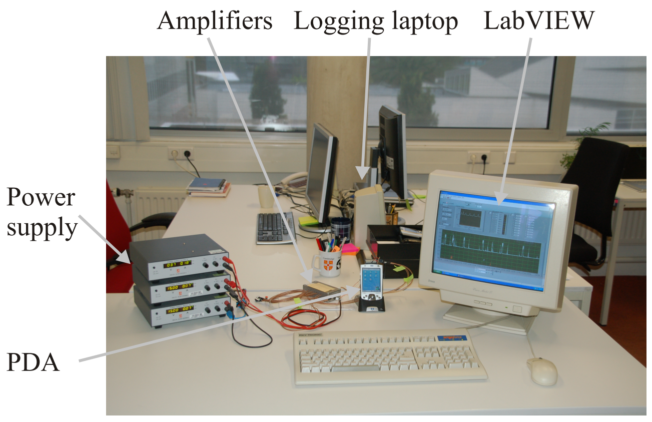
The method we adopt throughout this research is experimental: simulation based for MEMS-based storage and empirical for Flash. MEMS-based storage devices are not available for us, therefore we model a MEMS-based storage device. We adopt the MEMStore model from CMU, refine it, and update its settings to mimic the IBM MEMS device. For Flash memory, we build an experimental setup resembling a mobile environment. We experiment with a CompactFlash (CF) card, and measure its energy consumption. Furthermore, we record I/O traces to drive our MEMS model for a sound comparison with Flash in terms of timing performance and energy consumption.

3.1 Experiment setup

Our experiment setup has a hardware part and a software part. The hardware part consists of a Personal Digital Assistant (PDA), a measurement setup, and a logging computer. Figure 3.1a shows a picture of the whole setup. The PDA represents the mobile device, on which the experiments run. The measure-

Parts of this chapter are based on two joint technical reports between the Electrical Engineering and Computer Science departments of the University of Twente [Khatib: 7, 8].

(a) Picture



(b) Networking

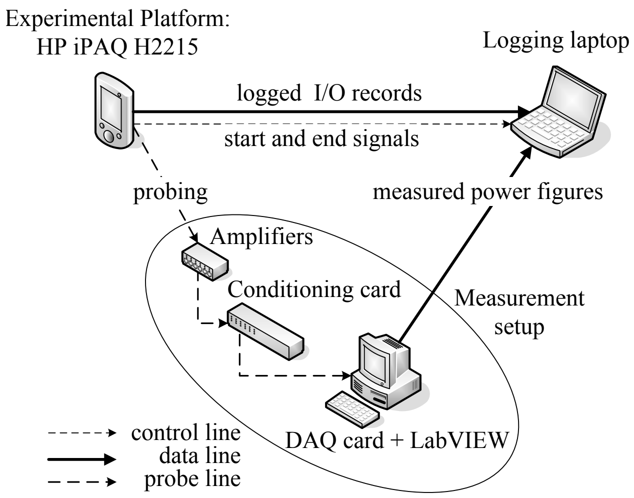


Figure 3.1: A picture and a sketch of the setup we used to experiment with Flash memory and to collect traces to drive simulations of the MEMS model.

ment setup is a personal computer equipped with a data acquisition card, and runs LabVIEW to collect readings of the power dissipation. LabVIEW is a platform from National Instruments, which is used for, among others, data acquisition and processing [57]. The logging laptop computer is used to collect the I/O requests from the PDA, and the measurement data from the measurement setup. Figure 3.1b shows a sketch of the networking of the setup.

Table 3.1: Specifications of our experimental setup: an HP iPAQ H2215 PDA

Component	Specification
Processor	Intel 400 MHz XScale PXA255
Memory	64 MB SDRAM
Display type	3.5-inch TFT active matrix
	16-bit (65536 colors)
	240 × 320
Expansion slots	1 SecureDigital (SD) Memory Card
	1 CompactFlash (CF) Card Type I/II
Distribution version	Familiar 0.83
Kernel version	Linux 2.6.16 with a block-tracing patch
GUI distribution	Open Palm Integrated Environment (OPIE)

3.1.1 The Experimental Platform

The platform on which our experiments run is an HP iPAQ H2215 Personal Digital Assistant (PDA), which we use to study capacity-modest applications. Table 3.1 lists its key specifications. The choice of a PDA as our experimental platform stems from our vision — at the beginning of our research — that PDA functionalities will continue to grow in handheld devices, and merge with the telephony functionality. The choice of this particular PDA model is twofold. Firstly, it is powered by a relatively modern mobile ARM-based processor: the Intel XScale PXA255. Secondly, the datasheet of its internal electronics are open, which allowed the Open Embedded (OE) community [58] to port full Linux onto it.

The battery of the PDA is replaced by a constant voltage supply throughout the experiments. The supply provides the PDA with a constant voltage (V_{dd}), so that to measure its the power dissipation it suffices to probe the change in the current only. By measuring the voltage drop (V_{R}) across a resistor (R), in series with the component under observation, we can calculate the power dissipated by the component (P_{c}) as follows:

$$P_{\mathrm{c}} = \frac{V_{\mathrm{R}}}{R} \cdot (V_{\mathrm{dd}} - V_{\mathrm{R}}).$$

The Flash memory we use comes in the form of a CompactFlash (CF) card that is plugged into the PDA across the so-called CF interface. The CF card is 2 GB in size, and is a Standard CF card from SanDisk [59]. The CF card functions as the main storage device on which the root file system is located. As a result, all I/O activities go from and to the CF card. We fix the transfer settings between the CF and the PDA throughout the experiments. Table 3.2 lists these settings.

Table 3.2: The transfer settings of the CF card applied throughout all experiments

Parameter	Setting
Transfer mode	PIO4 (120 ns or 16.7 MB/s)
I/O bus width	16 bit
Multi-count[a]	0 blocks (off)

[a] *Multi-count* is the number of blocks to be additionally serviced with the originally requested blocks.

We deploy a transfer mode with the highest transfer rate available, namely the Programmed Input/Output (PIO) mode 4, which supports a transfer rate of up to 16.7 MB/s on a 16-bit data bus.

To perform our experiments, we install a Linux distribution onto the PDA. The Linux distribution is called Familiar [60]. The Linux operating system gives us the freedom in implementing necessary tools, since it is open source. Further, it is rich in utilities, which enhances our productivity. We opt for the OPIE graphical user interface. OPIE encompasses a wide range of applications including streaming, web browsing, and office applications.

We patch the kernel with Jens Axboe's block-trace utility [61]. This utility enables us to log the I/O requests between the file system and the storage device. The output of the block-trace utility is sent over a TCP connection to the logging laptop, where I/O traces are collected for simulations. This avoids contaminating the gathered I/O traces by the operations needed to store trace records locally. Appendix A discusses an excerpt from one of our I/O traces.

3.1.2 The Measurement Setup

The measurement setup, which is on loan from Philips Research Laboratories in Eindhoven, consists of several components: a Data Acquisition (DAQ) card, NI 6040E; a connector board, BNC-2110; a custom-built amplification box; and a LabVIEW-based program. The NI 6040E DAQ card has 8 differential channels multiplexed to one 12-bit Analog-to-Digital Converter (ADC). We used three channels: one for the whole device, and the other two for the two supply pins of the CF card. In all experiments, the ADC sampling rate is set to 150 KS/s (i.e., kilo Sample per second). The conditioner is used for, among others, isolating high-voltage transients that could damage the computer. The LabVIEW-based program initializes the measurement setup, acquires samples, visualizes the samples, and sends them over a TCP connection to the logging computer for later analysis.

Using 1 Ω resistors, we measure the power dissipated by the storage device and the rest of the system separately. The voltage drops are in the order of millivolts. To be able to measure such voltages accurately, we use the amplification box to amplify the signals by a factor of 100. Furthermore, to increase the sensitivity for small voltage changes, we set the internal measurement range of the DAQ card to [-5,5] V, resulting in an additional amplification factor of four because the full range is 40 V. The smallest detectable voltage change depends on the least significant bit (LSB) of the 12-bit digital value, which is:

$$\frac{\text{voltage range}}{\text{amplification factor} \times 2^{\text{resolution}}} = \frac{5-(-5)}{4 \times 100 \times 2^{12}} = 6.1\mu V\,.$$

This value represents our measurement sensitivity, the least detectable change in voltage. After eliminating external sources of error, such as radiative and capacitive coupling, our measurement error due to internals of the DAQ card is $\pm 6.1\mu$V. Appendix A discusses an excerpt from one of the power traces we have collected.

3.2 Modeling MEMS-Based Storage

Our approach in studying MEMS-based storage devices is simulation based, since MEMS-based storage devices are not yet available to us. We adopt the MEMStore model from CMU [40], and refine it to approximate the IBM MEMS device. The MEMS model consists of a (timing) performance part and an energy part. This model can be integrated in the DiskSim simulation environment. DiskSim [41] is a configurable disk system simulator originally developed at the University of Michigan. DiskSim has been extended by researchers to encompass models for the SSD and MEMS-based storage.

The models are partitioned into parts with respect to the operation modes a MEMS-based storage device can operate in. A MEMS-based storage device can be in five different operation modes: (1) active, (2) seek, (3) idle, (4) shutdown, and (5) inactive (see Section 4.1.1). To ease the discussion of the models, we walk through each operation mode and discuss the corresponding performance and energy models, pointing out the operational component in each mode.

Our MEMS model captures the performance and energy consumption of a MEMS-based storage device in each of these operation modes. Our contribution to the performance model is mainly to the shutdown mode. On the other hand, we revise the energy model to account for electromagnetic actuators, whose power dissipation varies depending on the position of the medium. We provide energy models for all operation modes. Before we discuss the performance and energy models, we present their parameters first.

Table 3.3: Experimentally measured values of the parameters of the IBM MEMS device [10]

Parameter	Symbol	Value	Unit
Actuator resistor	R_{coil}	8.4	Ω
Maximum current	$i_{\max}$	0.2	A
Sled scan speed	v_{a}	0.001	m/s
X spring constant	k_{x}	104.0	N/m
Y spring constant	k_{y}	91.0	N/m
Sled mass on X	m_{x}	0.000102	kg
Sled mass on Y	m_{y}	0.000082	kg
X actuator force constant	n_{x}	0.062	N/A
Y actuator force constant	n_{y}	0.055	N/A
Resonance frequency on X	f_{ox}	161.0	Hz
Resonance frequency on Y	f_{oy}	168.0	Hz

Table 3.4: Settings of the DiskSim MEMS model obtained from and calculated based on the measured settings in Table 3.3. The table adopts the units used by the DiskSim MEMS model.

Parameter	Symbol	Value	Unit
Maximum sled movement on X	d_{x}	100.0	µm
Maximum sled movement on Y	d_{y}	100.0	µm
Bit cell length	d_{bit}	25.0	nm
Per-probe data rate	$r_{\text{probe}} = \frac{v_{\text{a}}}{d_{\text{bit}}}$	40.0	Kbps
Maximum acceleration on X	$a_{\text{x}} = \frac{F_{\text{x, max}}}{m_{\text{x}}} = \frac{i_{\max} \cdot n_{\text{x}}}{m_{\text{x}}}$	51.17	m/s^2
Maximum acceleration on Y	$a_{\text{y}} = \frac{F_{\text{y, max}}}{m_{\text{y}}} = \frac{i_{\max} \cdot n_{\text{y}}}{m_{\text{y}}}$	55.73	m/s^2
Resonance frequency		161.0	Hz
Spring constant factor[a]		–	
Settling time constant[a]		–	
Startup delay [b]		0.0	
Startup power [b]		0.0	
Inactive power		5.0	mW
Command overhead		0.1	ms

[a] These parameters pertain to the single-parameter seek model devised by Griffin *et al.* [40]. We do not set them, since we adopt Hong *et al.'s* [43] advanced seek model.

[b] MEMS-based storage devices have no startup overhead (see Section 4.1.1).

3.2.1 Model Parameters

We detail the parameters of the mass-spring-damper system that models the sled motion in a MEMS-based storage device. Table 3.3 lists the parameters along their settings, obtained from a recent prototype of the IBM MEMS device [10]. Table 3.4 lists the settings of the parameters of the DiskSim MEMS model. While some parameters of the MEMS model can be set directly, others are calculated as shown in Table 3.4.

3.2.2 Timing Performance Model

Schlosser [46] discusses the performance model in great detail. We briefly discuss it for each operation mode, and detail the shutdown mode.

1. Active mode In the active mode, a MEMS-based storage device reads and writes user data. The time it takes to read or write bits is determined by the (scanning) velocity the medium can travel at, while a probe can still manipulate bits. The scan velocity is constant during read/write (or access) operations. From the bit dimension and the access velocity, the per-probe data rate is calculated as shown in Table 3.4. Before a request is serviced by the sled, the (x, y) coordinate should be calculated based on the internal address mapping (as we explain in Section 3.2.4). We allocate a 0.1 ms controller overhead time for such processing as used in the DiskSim simulator.

2. Seek mode The sled accelerates and decelerates during a seek. Seek operations cannot be simulated accurately with a simple model that has no control loop, which makes it difficult to calculate the exact seek time. However, it is possible to calculate the best possible seek time from the maximum possible acceleration; this will give a lower bound on the seek time. This calculation assumes maximum actuator force — by applying the maximum allowed current — during the seek (first accelerating, then decelerating). In our case, this maximum force is constant, and can be calculated as shown in Table 3.4. This so-called “bang-bang” control is time optimal. Measurements performed by IBM indicate that the measured seek time of the used control loop is within 19% from the theoretical optimal seek time calculated by the bang-bang model [62].

The DiskSim MEMS model implements two seek models. Griffin *et al.* [40] calculate the seek time using a piecewise approximation of the bang-bang model. Hong *et al.* [43] devise an analytical solution to the bang-bang model. We adopt Hong *et al.’s* seek model. Because the movements along the X and Y directions are independent (recall characteristic M7 of MEMS-based storage devices), the seek time is the maximum of the X and Y seek times:

$$t_{\text{seek}} = \max(t_{\text{seek,x}}, t_{\text{seek,y}})\,. \tag{3.1}$$

3. Idle mode In idle mode, the sled moves at the scan speed over the last visited track anticipating requests in the neighborhood. Since the sled moves at a constant velocity, the performance model of the idle mode is that same as that of the transfer mode, but without read/write activity.

4. Shutdown mode We devise two models corresponding to two different shutdown policies. We call these policies a performance-efficient shutdown policy and an energy-efficient shutdown policy. We explain the policies in Section 4.3.

The shutdown time is modeled in a similar way as the seek time except that the sled comes to a full stop on the X and Y directions at the center. The two shutdown models differ in the force applied to accelerate the sled toward the center. While the model of the performance-efficient policy involves the actuators force to accelerate the sled in addition to the springs restoring force, the second model uses the springs force only. The complete analytical derivation of the models is offered in Appendix B. The shutdown time for either policy is:

$$t_{\text{shutdown}} = \max(t_{\text{shutdown,x}}, t_{\text{shutdown,y}})\,. \tag{3.2}$$

5. Inactive mode In the inactive mode, the sled is stationary at its resting position, called the center position, which is conventionally taken at the (0,0) coordinates. Unlike disk drives, a MEMS-based storage device has no startup overhead, since the sled is always suspended across the probe array (see Section 4.1.1). Therefore, we set the startup overhead to zero in the MEMS model as shown in Table 3.4.

3.2.3 Energy Model

We calculate the energy consumed by the actuators to hold the sled still in X and to move it in Y. Energy is calculated using the power dissipated at a certain position, say (x, y). We assume electromagnetic actuators, such as those deployed in the IBM MEMS device. The electromagnetic actuators are driven by a controlled current source. The current source provides the amount of current needed to put the actuator in equilibrium at a certain position, as given in Equation (3.3). We calculate for X actuators, while the same is applicable for Y actuators. The voltage across the current source is given in Equation (3.4). Since the device operates at a frequency well below the resonance frequency (10 Hz respectively 161 Hz), we can neglect the second term of Equation (3.4). Thus, we can approximate the power dissipation at position x as given in Equation (3.5). Similarly, the power dissipation at position y is given in Equation (3.6), assuming the coils of the Y actuators have the same resistor of X ones.

$$i(x) = \frac{k_\mathrm{x}}{n_\mathrm{x}} \cdot x \tag{3.3}$$

$$u(x) = i(x) \cdot R_\mathrm{coil} + n_\mathrm{x} \cdot \frac{dx}{dt} \tag{3.4}$$

$$P(x) \approx \frac{{k_\mathrm{x}}^2 \cdot R_\mathrm{coil}}{{n_\mathrm{x}}^2} \cdot x^2 \tag{3.5}$$

$$P(y) \approx \frac{{k_\mathrm{y}}^2 \cdot R_\mathrm{coil}}{{n_\mathrm{y}}^2} \cdot y^2 \tag{3.6}$$

1. Active mode In the active mode, the sled is stationary in the X direction, and moves along the Y direction. The total energy consumed for read/write is the sum of the energy to stay still in the X direction at position x, the energy to move along Y from y_1 to y_2, and the energy consumed by the probes and their associated electronics:

$$\begin{aligned} E_\mathrm{RW} &= E_\mathrm{actuation} + E_\mathrm{probes} \\ &= E_\mathrm{stay}(x) + E_\mathrm{move}(y_1 \to y_2) + E_\mathrm{probes}, \end{aligned} \tag{3.7}$$

where

$$E_\mathrm{stay}(x) = P(x) \cdot t_\mathrm{RW} = \frac{{k_\mathrm{x}}^2 \cdot R_\mathrm{coil}}{{n_\mathrm{x}}^2} \cdot x^2 \cdot t_\mathrm{RW}, \tag{3.8}$$

$$\begin{aligned} t_\mathrm{RW} &= \frac{y_2 - y_1}{v_\mathrm{a}} \\ y(t) &= y_1 + t \cdot v_\mathrm{a} \ , \quad t \in [0, t_\mathrm{RW}] \\ E_\mathrm{move}(y_1 \to y_2) &= \int_0^{t_\mathrm{RW}} P(y(t))dt \\ &= \int_0^{t_\mathrm{RW}} \frac{k_\mathrm{y}^2 \cdot R_\mathrm{coil}}{n_\mathrm{y}^2} \cdot y^2(t)\ dt \\ &= \frac{k_\mathrm{y}^2 \cdot R_\mathrm{coil}}{n_\mathrm{y}^2} \cdot \left[\frac{1}{3 \cdot v_\mathrm{a}} \cdot (y_1 + t \cdot v_\mathrm{a})^3 \right]_0^{t_\mathrm{RW}} \\ &= \frac{k_\mathrm{y}^2 \cdot R_\mathrm{coil}}{3 \cdot v_\mathrm{a} \cdot n_\mathrm{y}^2} \cdot \left(y_2^3 - y_1^3\right), \end{aligned} \tag{3.9}$$

where $y_2 > y_1$ and

$$E_\mathrm{probes} = N \cdot P_\mathrm{probe} \cdot t_\mathrm{RW}, \tag{3.10}$$

where N is the number of probes used for reading/writing, and P_probe is the power dissipated per probe (and its electronics) to read/write a single bit. The read/write time, t_RW, is calculated from the performance model.

2. Seek mode Recall that we adopt the bang-bang model, which drives the sled with maximum force. The maximum force is achieved by providing the actuators with the maximum allowed current, i_{max}. The seek energy is the sum of the energy consumed to seek along X and Y:

$$
\begin{aligned}
E_{\text{seek}} &= i^2_{\text{max}} \cdot R_{\text{coil}} \cdot (t_{\text{seek,x}} + t_{\text{seek,y}}) \\
&+ \begin{cases} \frac{k_{\text{x}}{}^2 \cdot R_{\text{coil}}}{n_{\text{x}}{}^2} \cdot x^2 \cdot (t_{\text{seek,y}} - t_{\text{seek,x}}) & ; \text{if } t_{\text{seek,y}} > t_{\text{seek,x}} \\ 0 & ; \text{if } t_{\text{seek,y}} = t_{\text{seek,x}} \\ \frac{k_{\text{y}}{}^2 \cdot R_{\text{coil}}}{n_{\text{y}}{}^2} \cdot y^2 \cdot (t_{\text{seek,x}} - t_{\text{seek,y}}) & ; \text{if } t_{\text{seek,y}} < t_{\text{seek,x}} \end{cases}
\end{aligned}
\tag{3.11}
$$

The second term of Equation (3.11) represents the energy consumed to stay still in one direction while completing the seek in the other direction. Equation (3.11) does not account for the settling energy, since a relatively small current is drained, and thus the power dissipation is negligible.

3. Idle mode The model of the energy consumption in the idle mode is the same as the one presented for the active mode except that no energy is consumed by the probes. Thus, Equation (3.7) holds for the idle mode after setting $E_{\text{probes}} = 0$.

4. Shutdown mode The performance-efficient shutdown policy uses the actuators to accelerate and decelerate the sled, consuming energy throughout the shutdown time. Therefore, when employing the performance-efficient shutdown policy the shutdown energy can be modeled as the seek energy except that no energy is consumed to stay still at the center. At the center, no counter force is needed. We refer the reader to Section 4.3 for further details on these policies. The shutdown energy becomes:

$$E_{\text{shutdown}} = i^2_{\text{max}} \cdot R_{\text{coil}} \cdot (t_{\text{shutdown,x}} + t_{\text{shutdown,y}}) \tag{3.12}$$

In contrast, the energy-efficient shutdown policy uses the actuators just for decelerating the sled to stop it at the center. Thus, energy is consumed during the deceleration time only.

$$E_{\text{shutdown}} = i^2_{\text{max}} \cdot R_{\text{coil}} \cdot (t_{\text{deceleration,x}} + t_{\text{deceleration,y}}) \tag{3.13}$$

Like in seek mode, in shutdown mode the actuators are used at maximum thrust (i.e., bang–bang) to shorten the shutdown time, explaining why i_{max} is present in Equation (3.12) and Equation (3.13).

5. Inactive mode In the inactive mode, the sled, the probes, and a large part of the electronics are switched off. The interface stays on anticipating requests from the system. A constant inactive power dissipation (P_{inactive}) is 5 mW, which is the inactive power measured on the CF card, assuming our simulated MEMS-based storage device is housed in a CF package. The inactive energy is:

$$E_{\text{inactive}} = P_{\text{inactive}} \cdot t_{\text{inactive}} \tag{3.14}$$

Since no startup overhead exists, a MEMS-based storage device consumes no startup energy when resuming from the inactive mode.

Next, we explain the data layout of MEMS-based storage devices, which we adopt throughout this dissertation.

3.2.4 Data Layout

The data layout is concerned with the way user data are organized on the storage medium of the storage device. We can split this organization into two parts: (1) low-level data layout, and (2) logical data layout. The low-level layout clusters bits into larger access and storage granularity, namely the sector. The logical data layout assigns a logical number to each sector on the medium in a particular way that enhances the performance of the storage device for a specific workload type. This section discusses the two parts.

The Low-level Data Layout

Probes in a MEMS-based storage device share one storage medium. As a consequence, the storage medium is logically divided into probe storage fields, each accessible by one single probe (characteristic M5 in Section 2.1.4). The size of the field is determined by the spacing between adjacent probes, which is the same for all probes (characteristic M4). Figure 3.2 shows an example storage medium that has $P \times Q$ probe fields (4×4 in this case), each contains $p \times q$ bits (8×8 bits in this case).

To find data on its moving medium, a MEMS-based storage device records marks on its medium *a priori*. These marks guide the sled's motion. They are called servo information. Designers of MEMS-based storage devices dedicate certain fields on the medium for servo information, called servo fields (not shown in Figure 3.2 for simplicity). As a result, servo information does not interleave with user data in every probe field, resulting in an increase in the effective capacity of the device. The IBM MEMS device has exclusive servo fields [63].

Probes are clustered into probe sets. A sector is striped across a probe set, so that each probe accesses a subsector in order to enhance the performance (characteristic M3). Figure 3.2 (right-hand part) shows the organization of user data when zooming in on one probe field. Error correction bits are added to user data (characteristic M11), and then stored in subsectors. Consecutive subsectors are separated by a few (overhead) bits to allow for buffering before writing a subsector, and to keep the read channel running to read an entire subsector [64, Chapter 18, pages 650 – 652].

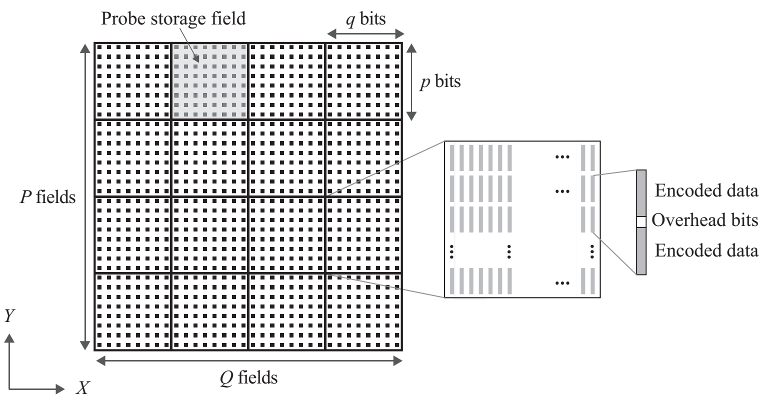


Figure 3.2: A look at the storage medium of a MEMS-based storage device split logically into $P \times Q$ storage fields, each is exclusively accessible by a single probe. Bits are clustered together into a sector. Each sector contains additional overhead bits for error correction, for example.

One of the main contributions of this work is taking a top-down approach in exploiting possibilities to enlarge the sector size and thus the subsector size. Using knowledge of the expected workload, we show that the subsector size can be enlarged above the minimum determined by the underlying recording technology. Enlarging the subsector size results in enhancement of the performance of the device, reduction in its energy consumption, and increase in the effective user capacity (Section 4.4).

MEMS-based storage devices are mechanical (characteristic M9), so that data are laid out on contiguous lines of bits, called tracks, along one direction. Throughout this work, we conventionally take the Y direction as the access direction as shown in Figure 3.2. Laying the data along one direction speeds up sequential accesses. An interruption to the Y motion occurs when the sled has to move along the X direction to change track, and to reverse the motion direction of the sled when reaching a track end.

The Logical Data Layout

The logical data layout assigns each sector a different Logical Block Address (LBA), with which a sector is uniquely addressable. Before detailing the address mapping, we briefly explain why the computer system addresses sectors with their respective LBAs instead of their physical addresses. In other words, we explain the reason behind the use of the block interface.

Block interface The complexity of storage devices has increased over time due to the increase in storage density and the subsequent increase in the number of housekeeping tasks. Today's storage devices perform almost all their housekeeping tasks, which the computer system was undertaking in the past. Such tasks are address mapping, and defect management, to name a few. Also, new tasks have been introduced, such as scheduling, power management, and cache management.

This takeover in tasks has resulted in a simplified interface between the storage subsystem and the computer system. Today, a storage device is interfaced as a sequence of blocks through the so-called *(logical) block interface*. A computer system accesses data on a storage device by specifying the address of the first block (or sector) and the number of consecutive blocks required. The address is called the Logical Block Address (LBA). The storage device translates the LBA into a Physical Block Address (PBA) based on its internal address-mapping scheme [64, Chapter 18, pages 655 – 656]. We explain address mapping in MEMS-based storage devices next.

Definitions Address mapping assigns each physical sector a logical number, so that a storage device is interfaced as a block device of a sequence of blocks. A MEMS-based storage device should maintain a physical layout in order to translate LBAs into PBAs, and thus locates sectors on the medium. Schlosser [46] suggests using the terminology of disk drives for MEMS-based storage devices. We adopt his definitions of the track and the cylinder. But we redefine the sector and introduce the subsector in order to keep the terminology consistent across disk drives and MEMS-based storage devices. The definitions are visually depicted in Figure 3.2 and Figure 3.3a. They are:

Probe set A probe set is the set of probes needed to access a full sector. Figure 3.3a shows a MEMS-based storage device with eight probe sets, each consisting of two probes. The sets are numbered column wise along the Y access direction as shown by the gray arrow.

Track A track is the set of all subsectors located at the same X offset in all columns that are simultaneously accessible. For example, in Figure 3.3a the first and second columns are simultaneously accessible by probe sets 0 through 3. In these columns, the bits at the same X offset make up a single track, because they are always accessed together.

Cylinder A cylinder is the set of all tracks located at the same X offset in all columns. The MEMS-based storage device in Figure 3.3a has four cylinders.

Sector A sector is the access granularity of a storage device. A request to the storage device demands a multiple of a sector (Figure 3.3b).

Subsector A subsector is the set of contiguous bits a probe accesses per sector. A sector is striped across several probes (Figure 3.3b).

Thus, Figure 3.3a shows a MEMS-based storage device of eight probes sets. The device has four cylinders, each consists of two tracks. Each track is phys-

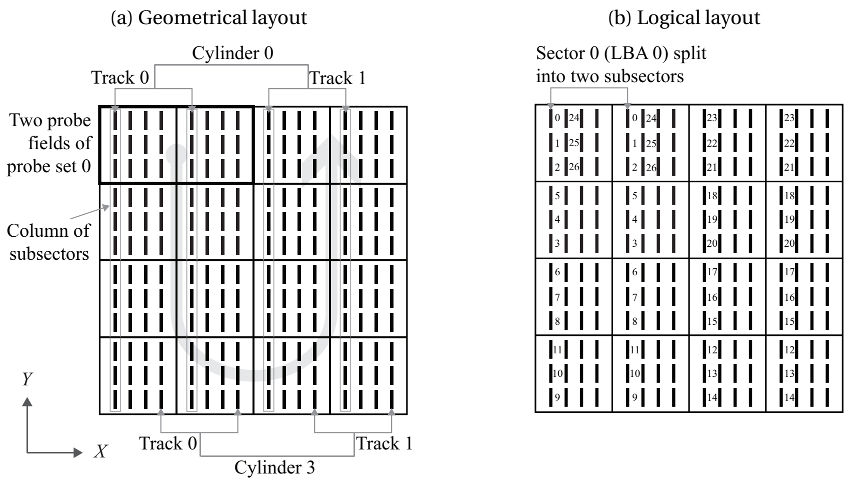


Figure 3.3: The geometrical and logical data layouts of the MEMS-based storage device of Figure 3.2. The geometrical layout borrows the sector, track, and cylinder terminology from the Disk drive. The logical data layout assigns consecutive numbers to contiguous sectors to exploit sequentiality. The depicted logical data layout follows the cylinder-mode address mapping scheme.

ically two columns of subsectors, that are simultaneously accessible by one probes set. Each column consists of 12 subsectors.

Block address mapping To optimize for sequential accesses, physically contiguous sectors on the storage medium should have consecutive Logical Block Addresses. Related data can be stored in these sectors, avoiding movements along the X direction. The movement along the X direction should be reduced, since it incurs a relatively long time to settle the probes over the center of a track. The settling time increases as the storage density increases.

Assigning logical numbers to sectors starts from sector 0 in track 0 of cylinder 0 and proceeds through all sectors of the track in the storage fields of probe set 0, such as sectors 0 – 2 shown in Figure 3.3b. Once the end of the fields of a set is reached, numbering continues to the next probe set. Always, when moving to the next probe set the sled reverses the direction, called **turnaround**, without moving along X. Numbering continues through the end of track 0 at sector 11. After that, numbering continues to the next track of the same cylinder, namely track 1 in this case. It does not change the probe set, and proceeds in the same way. Once a cylinder end is reached, sector 23 of cylinder 0, numbering continues to the next cylinder, which is cylinder 1, starting from its first track, such as sector 24. At every cylinder switching, a seek along the X is incurred. The numbering continues cylinder by cylinder.

This address-mapping scheme is called the cylinder mode in disk drives, because it numbers sectors in a cylinder-by-cylinder fashion. Another mapping scheme, called serpentine, numbers the sectors of the same track in all cylinders. Once the same track in all cylinders is numbered, it proceeds to the next track. For sequential accesses, the cylinder mode incurs fewer seeks along the X direction compared to the serpentine mode, resulting in better performance. We adopt the cylinder mode throughout this work.

3.3 Workloads

To drive the DiskSim MEMS model, we collect several I/O traces corresponding to different usage scenarios of the PDA. Before explaining each trace and its characteristic, we describe our tracing methodology first.

3.3.1 Tracing Methodology

Several measures should be taken into account when collecting I/O traces to warrant clean traces that exactly contain the I/O records pertaining to a certain system activity. Each of the traces we gather is recorded after a cold boot of the PDA to eliminate warm-cache effects. An I/O trace is captured for several sessions corresponding to different application and/or usage scenarios. After booting, a script launches sessions in a sequential fashion. Between every two successive sessions, the page cache and then the cache of the CF card are flushed to eliminate cross-session interference. Trace records are sent directly to the logging laptop to avoid contaminating the gathered traces by the operations needed to store trace records locally.

3.3.2 Traces

We captured three traces on the `ext3` file system formatted with the default maximum block size, 4 KB. The `ext3` [65] file system (or the third extended file system) is a journaling file system adopted as the default file system by many popular Linux distributions. The three traces are called: (1) `scenarios`, (2) `multimedia`, and (3) `iozone`. While the `scenarios` trace captures possible user activities, the `iozone` trace captures all possible access patterns a file system can exercise on a storage device. Each of the three traces was gathered for certain reasons pertaining to the type of study we carry out. Further, the `scenarios` trace was also captured when formatting with the `ext2` file system and a block size of 1 KB for conclusive results as discussed in Section 4.2.2. Table 3.5 summarizes the statistics of each trace. We plot the distribution of the seek distance, the request size, and the inter-arrival time of each trace in Appendix A.4. The traces have a different number of sessions that sufficiently cover various usage cases. This explains the difference in the number of I/O

Table 3.5: Statistics of the three traces used in this research

metric	`scenarios`	`multimedia`	`iozone`
Number of IOs	2679	21,867	410,627
Sequentiality percentage	44.1	82.7	21.2
Write percentage	56.4	40.4	43.2
Statistics of the request size [0.5 KB sector]			
Minimum	8.0	8.0	2.0
Median	8.0	8.0	8.0
Maximum	256.0	256.0	256.0
Mean	43.3	17.3	31.0
Standard deviation	69.3	27.8	54.2

requests between them. We also account for the delay between the I/O records and the corresponding power readings. These delays are incurred due to transportation over the network, which is depicted in Figure 3.1b. Next, we briefly discuss each of the three traces.

The `scenarios` trace The `scenarios` trace logs different system and application activities. System activities include booting and starting the graphical user interface. Application activities include launching applications, such as the text editor and web browser; copying files; and creating and deleting files. In addition, the trace contains streaming scenarios at 32 Kbps, 128 Kbps, and 384 Kbps corresponding to different audio and video qualities. Streaming at the three bit rates was carried out from and to the CF card, amounting to six streaming scenarios in total.

The `scenarios` trace contains I/O requests that are the result of a typical user activity on a handheld device. We believe this to be representative of I/O workloads in mobile systems. We use this trace mainly for our optimization studies in the next chapter.

The `multimedia` trace The `multimedia` trace logs photo taking, single and dual streaming from and to the storage device. Dual streaming represents a scenario where the user is playing back a stream and downloading another at the same time. All streaming scenarios are captured for audio (16 – 128 Kbps) and video qualities (64 – 2048 Kbps) with various chunk sizes (4 – 256 KB).

We use the `multimedia` trace mainly in the investigation of the suitability of MEMS-based storage devices in a predominately streaming environment. These studies are discussed in Section 6.3.

The* `iozone` *trace We ported the IOzone benchmark [66] to our PDA and experimented with several access patterns, including sequential, random and stride reads and writes with various record and stride sizes; a stride is the number of sectors separating two successive accesses. The wide coverage of access patterns of the IOzone benchmark enables us to simulate for almost every access pattern a MEMS-based storage device can encounter in reality, increasing the comprehensiveness of our study.

We use the `iozone` trace mainly to investigate access patterns that cause uneven wear of probes in MEMS-based storage devices. We devise several wear-leveling policies in Chapter 5, and test them against the `iozone` trace as well as the previous two traces.

3.4 Summary

This chapter presents our research methodology in tackling the integration problem of MEMS-based storage devices in the computer system. Our approach is experimental. We experiment with a CompactFlash (CF) card for Flash memory, whereas we simulate for MEMS-based storage. We approximate the prototyped IBM MEMS device in our simulations. We experiment on a mobile platform, and gather I/O traces to drive our MEMS model. We also measure the energy consumption of the CF card to compare it with the energy consumption of our modeled MEMS-based storage device. We take measures to collect clean I/O and power traces for meaningful results.

CHAPTER 4

Enhancement Policies

This chapter optimizes MEMS-based storage devices in order to prepare them for service in mobile battery-powered devices. Because of the limited battery capacity, a MEMS-based storage device must be energy- and performance-efficient. To keep its promised low cost per bit, a MEMS-based storage device must retain most of its physical capacity after formatting. We simulate against a model that is based on the IBM MEMS device.

We present three types of enhancement policies in this chapter. These are: the power management policy, the shutdown policy, and the data-layout policy. The first two address energy- and performance-efficiency, whereas the latter addresses the capacity additionally. We first devise the Power State Machine (PSM) of a MEMS-based storage device to optimize it systematically.

4.1 Operation Modes

Typically, a mechanical storage device can be in one of three different operation modes when it is switched on: (1) seek, (2) active, and (3) idle; we detail them shortly. Figure 4.1 presents the corresponding Power State Machine (PSM), and shows the transitions between the three modes. To reduce its

Parts of this chapter have been presented at the 2-nd International Workshop on Software Support for Portable Storage (IWSSPS'06), Seoul, Korea [Khatib: 5]; the 8-th ACM & IEEE International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'08), Atlanta, Georgia, USA [Khatib: 2]; and the 8-th ACM & IEEE International Conference on Embedded Software (EMSOFT'08), Atlanta, Georgia, USA [Khatib: 3].

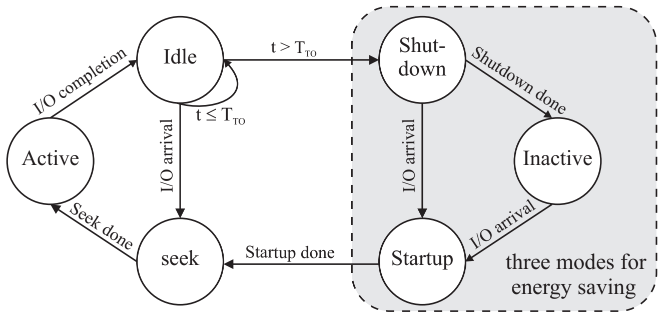


Figure 4.1: Three typical operation modes for a mechanical storage device extended by another three modes for energy saving

energy consumption, a mechanical storage device implements a low-power mode to halt the moving storage medium. As a result, the simple PSM is extended by another three operation modes as shown in the gray area of Figure 4.1. Transitioning between low-power modes and active modes goes via preparation modes, namely the startup and shutdown modes.

Like mechanical storage devices, a MEMS-based storage device consumes a significant amount of energy in the idle mode. Figure 4.2 gives the energy breakdown for various application sessions when simulating against the `scenarios` trace. It shows that at least 40% of the total energy is consumed during idleness. Therefore, a MEMS-based storage device should implement power management and have a PSM similar to the full PSM shown in Figure 4.1.

In the following, we construct the Power State Machine (PSM) of MEMS-based storage devices by (1) contrasting MEMS-based storage devices with HDDs, and (2) carefully inspecting their characteristics based on their state of the art. We answer these two questions:

- *How many low-power modes should the PSM have?*
- *What are the corresponding preparation modes?*

We answer these questions next, and evaluate our choice in the Section 4.2.

4.1.1 Contrasting with Hard Disk Drives

A Hard Disk Drive (HDD) has a PSM similar to that of Figure 4.1. A HDD can, however, implement several low-power modes of different rotation speeds [67] depending on the form factor and whether the HDD is commodity or enterprise equipment. Upon a request arrival, the disk enters the **seek mode** where the head seeks to the addressed track. Once the head is aligned over the right track, the head starts reading from or writing to the medium, referred to as the **active mode**. Upon completion of reading or writing data, the disk enters the **idle mode**, where its platters keep rotating in anticipation of further requests

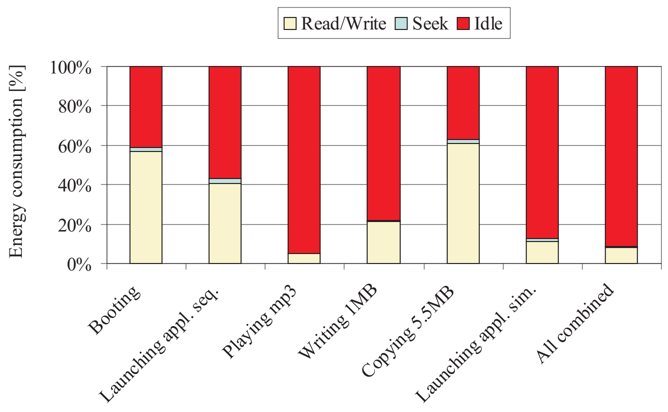


Figure 4.2: Energy breakdown for various applications of a MEMS-based storage device that has no power management. At least 40% of the total energy is consumed in the idle state.

shortly. If Power Management (PM) is employed, the disk transitions from idle mode to the **shutdown mode**, where it stops the medium, switches off a large part of the electronics, and then parks the heads. Upon shutdown completion, the disk is in **inactive mode**, where just its interface is on to awake the disk, when a request arrives. Upon request arrival, the disk goes in the startup mode. In the **startup mode**, the platters spin up and gain a certain speed first. After that, the heads can be loaded to fly over (and not touch) the platters separated by the air pad, which is created by the spinning platters [64, Chapter 17, pages 631 – 633]. This spinup activity takes several seconds depending on, among others, the mass of the platters and the target spinning speed. After spinup, the head seeks to the addressed track to satisfy the request and so on.

Unlike in a HDD, in a MEMS-based storage device, the media sled is suspended by springs across the probe array at a specific distance from the storage medium (see Figure 2.2), which is actively maintained by the Z nanopositioners (recall characteristics M1 and M8 from Section 2.1.4). As a result, *no startup overhead exists.* To access a MEMS-based storage device, that is in inactive mode, the media sled directly seeks along X and Y simultaneously to the addressed data block, since they can move independently at the same time (characteristic M7). As a result of the independent motions, the seek time is the maximum of the seek times along X and Y, and the difference between the seek times along X and Y loosely corresponds to the rotational latency in a HDD. A MEMS-based storage device has a light-weight sled (characteristic M2), exhibits a high storage density (characteristic M10), and has a micro scale. These characteristics enable the realization of a high-capacity storage device in a small form factor. Because there are many probes, the sweep area of one probe is relatively small (characteristic M5). As a consequence, seek times are short. At shutdown the sled moves to its resting position, namely the center of the probe field, and remains stationary. Like the seek time, the shutdown time is short, and thus shutdown energy is small.

In the IBM MEMS device, a large number of probes (64×64 probes) share one medium, reducing the storage field of an individual probe to 0.01 mm^2. The media sled weighs approximately 0.1 g. In the prototype of the IBM MEMS device, the recording density is 840 Gb/in^2 (see Table 2.2). In contrast, for a small disk drive, for instance the Toshiba 0.85-inch drive, the single platter weighs 1 g, the accessible platter space per head is approximately 366 mm^2, the storage density is 30 Gb/in^2 [68]. Note that the storage density of the disk drive has increased since the time of the 0.85-inch drive; modern hard disk drives exhibit a storage density of 250 Gb/in^2.

In disk drives, startup, shutdown, and seek energy and delay are incurred every time the disk shuts down and subsequently starts up. Unlike disk drives, MEMS-based storage devices have no startup overhead, but experience small seek and shutdown overheads. The small overheads motivate us to implement a PSM of one single low-power mode. We present the Power State Machine (PSM) of a MEMS-based storage device next.

4.1.2 Power State Machine

Figure 4.3 shows the proposed Power State Machine (PSM) for MEMS-based storage devices. It consists of five operation modes: seek, active, idle, shutdown, and inactive. We detail these modes and their power dissipation figures, which we adopt from a relatively recent prototype of the IBM MEMS device [10]. In the **seek mode**, the media sled moves from its current position to the starting position of the next request to service it. Seeking dissipates 336 mW per direction, amounting to 672 mW in total. The seek model is the bang-bang optimal time model which applies the maximum current possible (Section 3.2.3). In the **active mode**, the device accesses (i.e., reads or writes) data, where the sled dissipates 60 mW at most to move along the Y direction, and another 60 mW at most to stay still in the X direction, since electromagnetic actuators are used. Recall from Section 3.2.3 that the power varies depending on the sled position, where peak power is dissipated at maximum displacement to counteract the spring forces. The 4096 probes and the error-correction electronics dissipate 1 W to read/write and correct data (approximately 0.25 mW per probe and its electronics). In the **idle mode**, the device anticipates requests to nearby locations. The sled moves along Y at the read/write velocity, while staying still in X, dissipating a maximum power of 120 mW in total. Upon request arrival, the device goes back into the seek mode. Otherwise, if within a certain time interval, that is equal to the time-out (T_{TO}), no request arrives, the device goes into the shutdown mode. In the

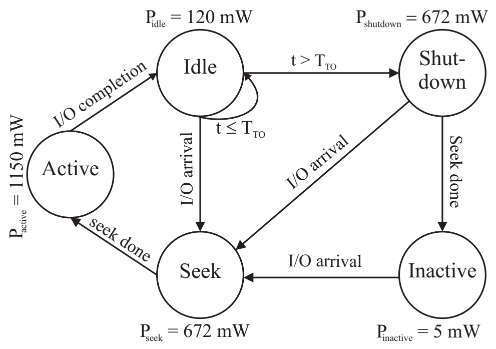


Figure 4.3: The devised Power State Machine (PSM) of MEMS-based storage devices. The power figure of the idle and active modes represent the peak power dissipated at maximum displacement.

shutdown mode, the sled travels to the center (resting) position from its current position, which is a seek operation but to the center. The actual travel time of the seeks depends on the distance. If a request arrives while shutting down, the device goes into the seek mode immediately. Otherwise, if no request arrives and the sled reaches the center position, the device goes into the inactive mode. In the **inactive mode**, the sled rests in the center position, the probes are switched off, and the interface awaits requests, dissipating 5 mW. No startup mode exists as explained previously, assuming the time to start up the electronics very short, so that it can be neglected.

4.1.3 Organization

Having detailed the PSM of MEMS-based storage devices, the next three sections optimize in three operation modes: (1) idle, (2) shutdown, and (3) active. In the following, we devise policies for the three modes.

Each of these policies is parameterized. We assess their influence for their optimal settings in the view of the principal design targets: energy consumption and response time. These policies are apparent in the PSM of Figure 4.3 and they are:

- The *power management policy* decides the time instance *when* to transition from the idle state to the shutdown state by setting the value of the timeout. We discuss it in Section 4.2.
- The *shutdown policy* decides on the way *how* to transition from the shutdown state to the inactive state. We discuss it in Section 4.3.
- The *data-layout policy* decides on the way user data are organized and thus *how* to process incoming requests. Consequently, the policy influ-

ences the state transition from the seek state to the active state and from the active state to the idle state. We discuss it in Section 4.4.

One state transition is left uncovered in the PSM by the above policies: from the inactive state to the seek state. The time spent in the inactive state is implicitly predefined by the workload exercised on the device.

We evaluate the design space of each policy systematically. Parameters are investigated using trace-driven simulations. The outcome of each study can be considered as a design rule, the designer of MEMS-based storage device can make use of. Using two real-world traces (the `scenarios` and `multimedia` traces) and applying our derived design rules, we compare a MEMS-based storage device to the CF Flash card in Chapter 6.

4.2 Power Management Policy

A MEMS-based storage device consumes a large amount of energy in the idle mode (see Figure 4.2). Therefore, to save energy the sled should be shut down, if the request queue is empty. The Power Management (PM) policy decides when to shut down the sled.

4.2.1 Fixed-Timeout Power Management

A large body of work on Power Management (PM) policies for disk drives and processors exists; Benini *et al.* [69] give a survey. Generally, there are two types of policies: static and dynamic policies. Dynamic policies achieve more savings at lower timing performance degradation than static policies [70]. However, dynamic policies demand more processing and memory resources, because they keep a history of recent timeout values and power states. Both types of policies can be also applied to MEMS-based storage devices.

The timeout (T_{TO}) in the Power State Machine (PSM) determines the time of the state transition from idle to shutdown. We parameterize the timeout to study the influence on performance and energy saving. By driving the PSM with real-world traces and changing the value of the timeout, we quantify the energy saving and the resulting performance of the device. As shown later, the quantification reveals the near-optimality of the fixed-timeout PM policy.

Resting the probe in the center position during inactivity increases the seek distance for the next request, if the request addresses the same region of the previous request. This is the case for real-world workloads, which exhibit spatial locality of reference (i.e., consecutive requests address nearby locations). A long seek distance results in a long seek time and thus a long response time. As a result, energy saving must be *traded off* for response time (i.e., performance). The tuning parameter of the energy–performance trade-off is the timeout. In general, the larger the timeout, the less the energy saving and the better the performance, and vice versa. We present the simulation results in Section 4.2.3.

4.2.2 Experimental Methodology

Recall from Chapter 3 that we adopt the settings of the IBM MEMS device, because of its maturity. We simulate using the model presented in Section 3.2, which approximates the IBM MEMS device. The simulation is driven by the `scenarios` trace, which we captured on the CF card when formatted with the `ext3` file system and a block size of 4 KB (see Section 3.3). We adopt the Power State Machine (PSM) presented previously, and set the timeout (T_{TO}) with incremental integral values. The timeout values are 0 – 10 ms. We, then, increase the timeout to 50 ms in steps of 10 ms to pursue the trend of the energy–performance trade-off.

MEMS-based storage devices will communicate with the file system layer in computer systems. As a result, the performance and energy consumption of MEMS-based storage devices is influenced by the type of the file system and its block size. For stronger conclusions, we further trace and simulate with different settings of the I/O subsystem. We captured the `scenarios` workload on the `ext2` file system, a non-journaling version of `ext3`. In addition, we formatted each file system with the default maximum block size, 4 KB, and a smaller size of 1 KB. Thus, we carry out simulations against four different traces of the `scenarios` workload: (1) `ext3-4K`, (2) `ext2-4K`, (3) `ext3-1K`, and (4) `ext2-1K`. Note that although these traces are induced by the same application scenarios, they differ due to their respective I/O settings.

We do not simulate against the `multimedia` and `iozone` traces (see Section 3.3) in the study of the Power Management (PM) and the shutdown policy in the next sections. The reason is that the `multimedia` trace represents streaming applications, which are inherently predictable, so that the optimal PM has a zero timeout. On the other hand, the `iozone` is not realistic in its arrival-time property, which is intrinsically relevant to PM. The reason is that the trace is generated by a benchmark that stresses out the file system and the storage device in a short period of time with various access patterns, such as read or write, and sequential or random.

Recall from Section 3.1 that the CF card is 2 GB in size, whereas the IBM MEMS device is approximately 7.6 GB at a bit pitch of 25 nm. In our simulations, we enlarge the bit pitch to 40 nm, while keeping the per-probe data rate the same at 40 Kbps. The resultant capacity is about 2 GB after taking the ECC codes and other overheads into account. We do so in order to ensure that requests to the device span the whole address space of the device. Thus, we always report the seeks that span the entire physical space of the device. Next, we present and discuss the simulation results.

4.2.3 Simulation Results

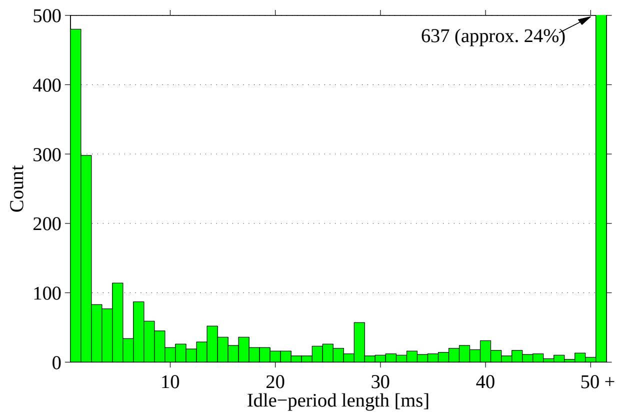
In this section, we discuss in detail the simulation results. The results present the trade-off between the energy consumption and the response time (i.e., the timing performance) for different values of the timeout. We first discuss the trade-off when simulating against the whole `ext3-4K` trace (i.e., all sessions combined). After that, we present the results for the other three traces (`ext3-1K`, `ext2-4K`, and `ext2-1K`), when simulating against each one as a whole. Last, we discuss the results for each session individually.

Whole `ext3-4K` trace

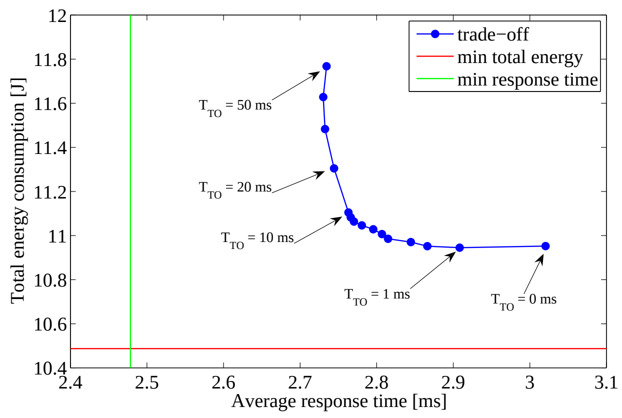
In general, the energy saving increases as the timeout (T_{TO}) decreases, because more idle periods are exploited to shut down the device. On the other hand, response time increases as the timeout decreases, because longer seek distances are incurred. Figure 4.4a and Figure 4.4b show the histogram of idle-period length and the energy–performance trade-off, respectively, when simulating against the whole `ext3-4K` trace. About 48% of the idle periods lie in the range of 0 – 10 ms.

Figure 4.4b plots the total energy consumption for the whole trace versus the average response time per request. It shows a decrease of 0.3 ms (approximately 10%) in average response time between $T_{\mathrm{TO}} = 0$ ms and $T_{\mathrm{TO}} = 10$ ms. This decrease is explained in Figure 4.4a, where the size of this decrease is proportional to the sum of the heights of the first ten buckets. When $T_{\mathrm{TO}} = 10$ ms, the first ten buckets are not exploited, avoiding longer seeks from the center position. In general, these buckets mainly influence the performance, because of their relatively high occurrence frequency. As the timeout increases, less performance degradation is incurred, because of the infrequent occurrence of long idle periods. We also observe an increase in response time at $T_{\mathrm{TO}} = 50$ ms relative to $T_{\mathrm{TO}} = 40$ ms. This is because the distance between the sled position and the position of the next request is not monotonic with respect to time. That is, it increases and decreases depending on whether the sled has reached its maximum displacement and then reversed its motion direction.

Figure 4.4b shows a slight decrease in energy consumption at $T_{\mathrm{TO}} = 1$ ms compared to $T_{\mathrm{TO}} = 0$ ms. The longer seeks explained above worsen not only performance, but also cost extra energy, so that avoiding the first bucket is more profitable than exploiting it. Also, no pronounced difference (approximately 2%) in energy consumption is seen in the range $T_{\mathrm{TO}} = 1 - 10$ ms, because of the small energy-saving contribution of their respective buckets compared to that of idle periods longer than 10 ms. From an energy-saving perspective, an occurrence of one idle period of length of 30 ms, for example, is more profitable than 30 occurrences of an idle period of length of 1 ms.



(a) Idle-period distribution



(b) Energy–performance trade-off

Figure 4.4: (a) The distribution of the idle-period length; There exist short (but many), and long (but few) idle periods. (b) Total energy consumption versus average response time when simulating against the whole `ext3-4K` trace; Timeout values 1 – 10 ms make little difference in energy consumption (approximately 2%) but vary in response time (approximately 10%). We show the minimum energy consumption and the minimum response time.

Optimality To quantify the optimality of the fixed-timeout PM policy with respect to energy saving, we calculate the **minimum energy consumption** (Figure 4.4b): the sum of (1) the energy consumed in reading or writing data; (2) the energy consumed in seeking, which are caused by requests only and not by PM; and (3) the energy consumed in inactivity. Figure 4.4b shows that the energy consumption at $T_{\mathrm{TO}} = 10$ ms is within 6% of the minimum (approximately a 0.6 J absolute difference). We also calculate the **minimum response time** (Figure 4.4b): the sum of (1) the transfer time, and (2) the seek time due to requests only and not due to PM. Figure 4.4b shows that the difference in response time at $T_{\mathrm{TO}} = 10$ ms is within 12% of the minimum (approximately 0.3 ms absolute difference).

However, we see that 8% out of this 12% can not be achieved by any value of the timeout (see $T_{\mathrm{TO}} > 10$ ms). The reason is that when the sled is left idle for some time it travels some distance that should be traveled back, if a successive request addresses the same neighborhood. Thus, in mechanical devices there is always a minimum seek distance that is incurred depending on the timeout. $T_{\mathrm{TO}} = 10$ ms, for example, is (near) optimal, since a room of just 4% is left for optimization on response time.

Other Traces

We repeat the simulation for the other traces taken for different I/O settings. These are `ext3-1K`, `ext2-4K`, and `ext2-1K`. The simulation results of each trace as a whole agree with the results of the `ext3-4K` trace discussed in the previous section. Table 4.1 presents the trade-offs for the simulated values of the timeout for each trace. For all traces, we observe that the actual minimum energy consumption is achieved when the timeout is in the range 1 – 10 ms. Another important observation is that increasing the timeout in the range 1 – 10 ms decreases the response time by a larger factor than it increases the energy. For example, setting the timeout to 10 ms instead of 1 ms decreases the response time by 7% at a 1% increase in energy consumption for the `ext3-4K` trace.

MEMS-based storage devices do not have startup energy, so that the relative difference between the energy consumption when $T_{\mathrm{TO}} = 10$ ms and the minimum does not exceed 7%, leaving very little room for improvement to dynamic policies. Thus, deploying a fixed-timeout Power Management (PM) policy with a PSM of one low-power mode is sufficient to achieve a near-optimal energy saving at low performance degradation.

Sessions Separately

As mentioned earlier, our traces consist of several sessions. The sessions include (1) booting Linux and starting the graphical user interface, (2) launching several applications sequentially, (3) playing an MP3, (4) writing a picture, (5) streaming from/to the storage device with different bit rates, (6) copying files and directories, and (7) launching several applications simultaneously. Since in reality not necessarily all of these sessions occur together, we split the traces into their respective sessions and simulate for each individual session.

The simulation results of each individual session for all traces agree with those of the entire traces. Figure 4.5a, Figure 4.5b, and Figure 4.8b show the energy–performance trade-off for sessions 4, 7, and 6, respectively. We observe that setting the timeout larger than 0 decreases the response time signif-

Table 4.1: Energy consumption versus response time for the four traces

T_{TO} [ms]	Energy consumption [J] / Response time [ms]							
	`ext3-4K`		`ext3-1K`		`ext2-4K`		`ext2-1K`	
0	11.0	3.0	10.4	1.8	9.9	3.3	9.7	2.6
1	10.9	2.9	10.3	1.7	9.9	3.2	9.7	2.6
2	11.0	2.9	10.4	1.7	9.9	3.2	9.7	2.6
3	11.0	2.8	10.4	1.7	9.9	3.2	9.7	2.6
4	11.0	2.8	10.4	1.6	9.9	3.2	9.8	2.6
5	11.0	2.8	10.4	1.6	9.9	3.2	9.8	2.5
10	11.1	2.8	10.5	1.6	10.0	3.2	9.8	2.5
20	11.3	2.7	10.7	1.6	10.0	3.2	9.9	2.5
30	11.5	2.7	10.9	1.6	10.1	3.2	10.0	2.5
40	11.6	2.7	11.1	1.6	10.1	3.2	10.1	2.5
50	11.8	2.7	11.2	1.6	10.1	3.2	10.1	2.5
Minimum	10.5	2.5	9.8	1.4	9.6	2.8	9.4	2.2

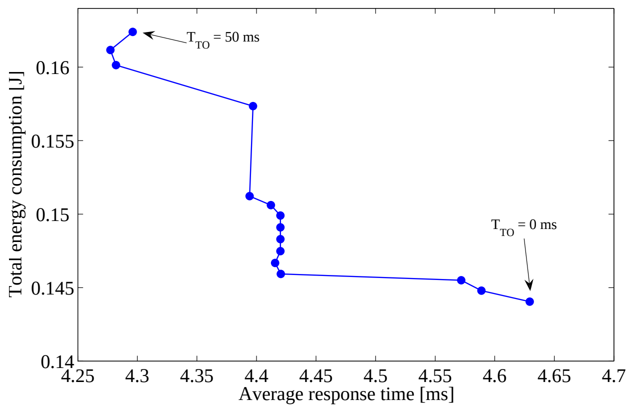
icantly for a relatively small increase in energy consumption, if any. Further, in conformity to the conclusion for the whole trace, the fixed-time PM policy achieves near optimal energy saving. If a workload exhibits a large percentage of idle periods that are shorter than 1 ms, setting the timeout to 0 worsens not only the response time, but also the energy consumption, since the shutdown energy becomes prominent. This is exactly the case for session 6 shown in Figure 4.8b.

4.3 Shutdown Policy

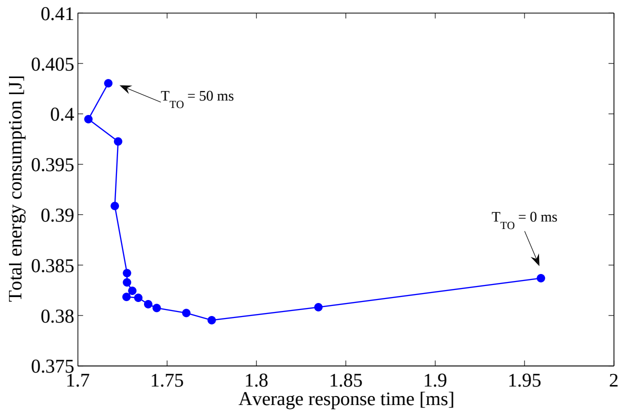
The study in the previous section demonstrates the possibility to shut down MEMS-based storage devices aggressively. Aggressive shutdown decisions are achieved by lowering the timeout, so that more idle periods can be exploited.

Decreasing the timeout, however, increases the number of shutdowns, and thus increases the shutdown energy. As a consequence, aggressive shutdowns can result in more energy consumption. This section exploits the architecture of a MEMS-based storage device to reduce its shutdown energy.

This section addresses the shutdown policy that complements the Power Management (PM) policy. While the power management policy decides *when* to shutdown, the shutdown policy decides *how* to shutdown. Two shutdown policies are possible; we detail them next.



(a) Taking a picture



(b) Firing applications simultaneously

Figure 4.5: Energy–performance trade-offs for two application scenarios from the `scenarios` trace when formatted with `ext3` and a 4 KB block size

4.3.1 Two Shutdown Policies

Shutting down means moving the sled to the center position. The (first) shutdown policy, we used in the previous section, employs the actuators to move the sled to its center position. Figure 4.6a illustrates that the actuators (F_a) in addition to the springs (F_s) exert a force on the sled and accelerate it for some distance. After that, the actuators reverse the force to decelerate the sled, so that it stops at the center. The actuators consume energy during acceleration and deceleration. The invested energy shortens the shutdown time, and therefore we call this policy the **performance-efficient shutdown policy**. We employed this policy in the study of the power management in the previous section.

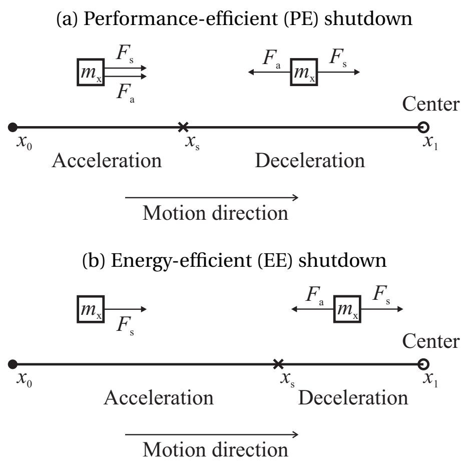


Figure 4.6: A sketch of the sled motion toward the center when shutting down with the performance-efficient and energy-efficient shutdown policies

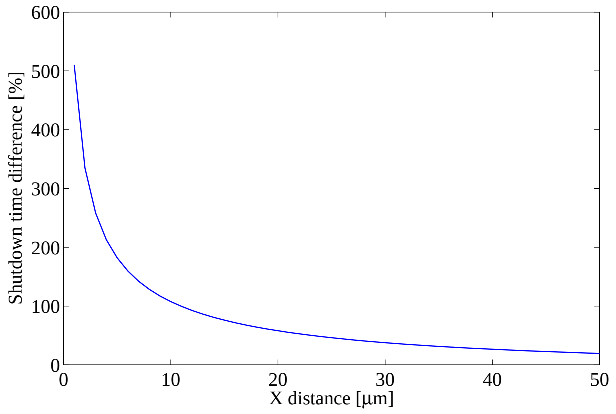
A second shutdown policy uses the potential energy stored in the springs only. The springs bring the sled as close as possible to the center (Figure 4.6b) before the actuators starts to decelerate the sled, so that it stops at the center. This policy consumes less energy than the previous policy, since it uses the actuators during deceleration only as Figure 4.6b illustrates. Therefore, we call it the **energy-efficient shutdown policy**. The energy benefit, however, comes at a performance cost. That is, the sled takes longer to reach the center, since it is not accelerated by the actuators.

We devise performance and energy models for the energy-efficient (EE) and performance-efficient (PE) shutdown policies. Appendix B offers analytical models of the shutdown time and energy for both policies. We implement these models in the DiskSim MEMS model to compare them under real-world traces. We also study the interaction between the PM policy and the shutdown policy. For better understanding of the behavior of the two policies, we provide an analytical (static) study first. After that, we follow up with a trace-based (dynamic) simulation that compares both policies in real environments.

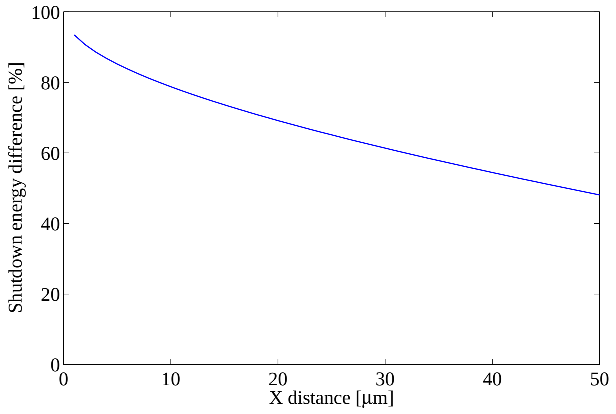
4.3.2 Analytical Study

The analytical study compares both policies when shutting down from every position within a probe storage field ($100\,\mu\text{m} \times 100\,\mu\text{m}$). We compare both policies with respect to the shutdown time and energy. The parameters of the models of both policies are set to the state-of-the-art figures from the IBM MEMS device [10]. The resting position is the center at the coordinate $(0,0)$.

Since the motions along the X and Y directions are independent and simi-



(a) Relative difference in shutdown time



(b) Relative difference in shutdown energy

Figure 4.7: Relative difference in shutdown time and energy between the performance-efficient (PE) and energy-efficient (EE) policies as a function of the distance from the center at 0. The difference is calculated as $\frac{t_{EE}-t_{PE}}{t_{PE}}$ and $\frac{E_{PE}-E_{EE}}{E_{PE}}$, respectively. PE outperforms EE, but consumes more energy.

lar, we present the results for the X direction only. The difference is calculated as $\frac{t_{EE}-t_{PE}}{t_{PE}}$ and $\frac{E_{PE}-E_{EE}}{E_{PE}}$ for the shutdown time and shutdown energy, respectively. The shutdown time and energy of the EE policy are referred to as t_{EE} and E_{EE}, respectively. We normalize the figures to the performance-efficient policy to demonstrate the relative gain in energy saving of the energy-efficient policy and the relative cost of the gain in terms of timing performance degradation.

Difference in Shutdown Time

Figure 4.7a shows that the relative difference in shutdown time (calculated as $\frac{t_{EE}-t_{PE}}{t_{PE}}$) is minimal at the borders of the probe area. It increases as the starting position gets closer to the center, since around the center the spring force is small. When deploying the EE policy, the sled accelerates to the center by the spring force only. The spring force depends on the distance between the sled's current position and the center ($F_S(x) \propto x$). That is the larger the distance, the more force but also the longer the distance the sled has to travel. As a result, shutdown times for all positions in the probe area are of the same order of magnitude; For example, the shutdown time at a 5 μm and 45 μm distance (in a range of 0 – 50 μm) is 1.6 ms and 2.0 ms, respectively.

In contrast, when deploying the PE policy the shutdown time scales very sensitively with the traveled distance, because it is actively accelerated; for example, the shutdown time at a 5 μm and 45 μm distance is 0.6 ms and 1.6 ms, respectively. This explains why the difference between both policies increases as the starting position gets closer to the center. The difference nearby the center is orders of magnitude longer than that at the borders, explaining the prohibitive large relative difference.

Difference in Shutdown Energy

Figure 4.7b plots the relative difference in energy consumption (calculated as $\frac{E_{PE}-E_{EE}}{E_{PE}}$). The shutdown energy when deploying the PE policy is larger than when deploying the EE policy, because the former consumes energy for acceleration and deceleration, whereas the latter consumes energy for deceleration only. Similar to shutdown time, the relative energy difference is larger around the center and decreases as the starting position gets further from the center. When deploying the EE policy, the long acceleration phase, which is responsible for the prohibitive difference in shutdown time, consumes no energy. Therefore, no prohibitive energy difference exists around the center, unlike the shutdown time.

Discussion

The analytical study shows that the energy-efficient shutdown policy saves more energy than the PE policy. However, the EE policy shows worse timing performance than the PE policy. The difference is prohibitively large nearby the center and can reach up to 500%, because of the (extremely) slow response of the EE policy in reaching the center.

The slow shutdown performance can be of advantage to real-world applications for two reasons. Firstly, from a cost viewpoint, shutdown is an overhead and not an inherent task of accessing data. That means it should be done as cheap as possible and not as quickly as possible. Secondly, the slow motion benefits those applications that exhibit (high) sequentiality and/or locality of reference. That is, moving the sled slowly allows for quick inexpensive seeks to an already visited region, if new requests demand further data from that region. In fact, we observe this advantage in our simulations presented next.

4.3.3 Trace-Based Study

We implement the models of both policies (see Appendix B) in DiskSim to evaluate the performance and energy influence of each shutdown policy in combination with the power management policy. We repeat the experiments discussed in Section 4.2.3 for all traces with the energy-efficient shutdown policy under real-world traces.

Whole `ext3-4K` trace

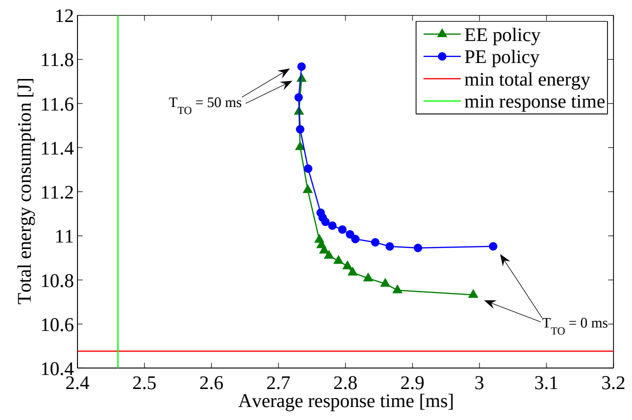
Figure 4.8a plots the trade-off between response time and energy consumption when deploying the EE shutdown policy and performance-efficient policy for the `ext3-4K` trace. We see that the energy decreases and becomes even closer to the minimum, further supporting the effectiveness of the fixed-timeout power management policy. At $T_{\mathrm{TO}} = 0$ ms, with the EE policy a MEMS-based storage device consumes less energy than with the PE policy, since shutdown energy (i.e., the overhead) becomes smaller.

Another finding is that the EE policy slightly shortens the response time compared to the PE policy, and thus enhances the performance. The reason is that for sequential requests the EE policy shortens the response time, because it moves the sled slowly to the center. As a result, driving the sled back to a previously visited region takes a small amount of time and thus consumes a small amount of energy. The difference is noticeable for the small values of the timeout, because of the large occurrence frequency of small idle periods as shown in Figure 4.4a.

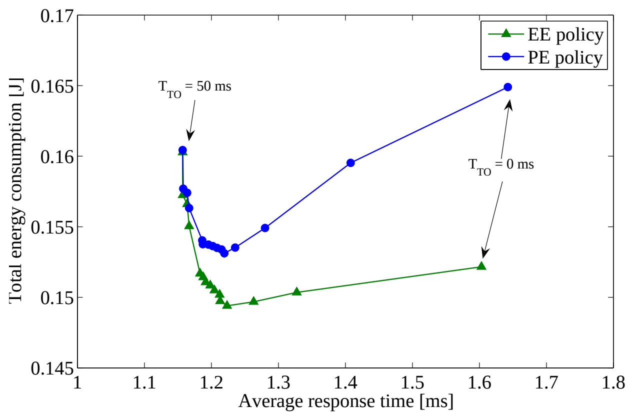
The energy saving difference between the two shutdown policies is negligible (approximately 2%) for the whole `ext3-4K` trace as shown in Figure 4.8a. This is because a quarter of the idle periods are longer than 50 ms, so that their corresponding energy saving outweighs savings corresponding to periods smaller than 1 ms. Note that for traces that lack such long idle periods due to a high arrival rate, in server systems for example, the difference in energy saving can be large. This is the case for the copying scenario, where the difference is approximately 10% as Figure 4.8b shows.

Figure 4.8a also shows that even with large values of the timeout, the minimum response time can not be achieved. The reason is that when the sled is left idle for some time it travels some distance that should be traveled back, if a successive request addresses the same neighborhood. Thus, in mechanical devices there is always a minimum seek distance that is incurred depending on the timeout.

Overall, we can conclude that the timeout $T_{\mathrm{TO}} = 10$ ms achieves a near-optimal energy saving (approximately 95%) at a negligible performance loss (4%), relative to the actual minimum represented by the line that connects all the points for $T_{\mathrm{TO}} \geq 10$ ms in Figure 4.8a.



(a) Whole `ext3-4K` trace



(b) Copying scenario

Figure 4.8: Energy–performance trade-offs when deploying the performance-efficient (PE) shutdown policy and the energy-efficient (EE) policy.

Other Traces

We repeat the simulation for the other three traces with the energy-efficient shutdown policy. We observe the same trend as for the `ext3-4K` trace. Our conclusion to deploy a timeout in the range of $1-10$ ms still holds, supporting the previous conclusions.

4.4 Data-Layout Policy

Data layout is concerned with the way user data are organized on the storage medium of a storage device. Data layout thus influences the timing performance and energy-efficiency of the storage device. For example, placing related data sectors contiguously on the physical medium avoids seeks between the sectors, which results in short response time and low energy to access data.

The attainable data rate per probe in a MEMS-based storage device is limited by several factors including the probe resonance frequency (characteristic M3 in Section 2.1.4). The per-probe data rate is 40 Kbps in the IBM MEMS device [10], suggesting that systems requiring even moderate transfer rates must use many parallel probes.

Because MEMS-based storage devices are mechanical, they must separate physical subsectors by gaps and embed a flush pad in each subsector to enable accessing stored data. Therefore, a small subsector size relative to the physical overhead results in a significant loss in capacity. As a result, the data layout also influences the capacity of a MEMS-based storage device.

The subsector size is determined by the striping policy, which has three parameters. We detail them next.

4.4.1 Three Data-Layout Parameters

A MEMS-based storage device uses many parallel probes. As a consequence, the data-layout design space widens beyond just block mapping, posing three questions that must be answered to maximize the timing performance and minimize energy usage without compromising capacity, namely:

1. **Total number of active probes (*N*):** how many probes should operate (i.e., be active) simultaneously?
2. **Sector parallelism (*M*):** how many sectors should be simultaneously accessible from the device?
3. **Sector size ($S_{\textbf{sector}}$):** should the conventional sector size of 512 bytes stay the same in MEMS-based storage devices?

The straightforward answers to these questions would be to (1) operate all probes simultaneously to gain peak throughput, (2) access one sector at a time to maximize bandwidth utilization, and (3) keep the sector size intact to access useful data only.

While these answers are logical, our research shows that none of the three targets (i.e., energy, performance, and capacity) of MEMS-based storage devices reaches optimality with a such configuration. Before studying the influence of each parameter on the design targets, we give an anatomy of the physical subsector first.

4.4.2 The Physical Subsector

A storage device stores user data in physical sectors. In addition to user data, a physical sector contains Error-Correction Code (ECC) data. All types of storage devices have to store ECC data to increase the reliability of the stored user data. The amount of ECC data depends on, among others, the sector size and

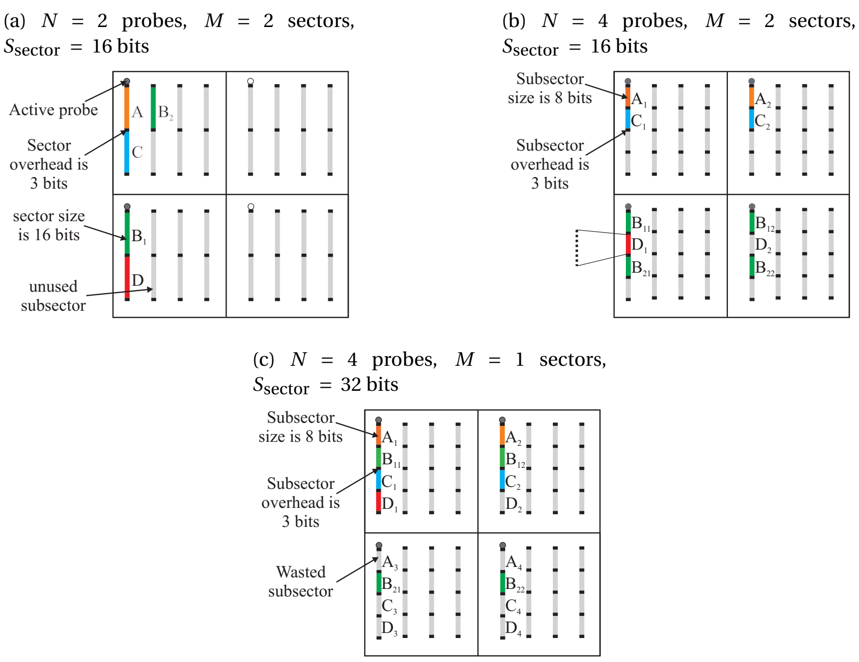

(a) N = 2 probes, M = 2 sectors, S_sector = 16 bits
Active probe
Sector overhead is 3 bits
sector size is 16 bits
unused subsector
A
B2
C
B1
D
(b) N = 4 probes, M = 2 sectors, S_sector = 16 bits
Subsector size is 8 bits
Subsector overhead is 3 bits
A1
C1
A2
C2
B11
D1
B21
B12
D2
B22
(c) N = 4 probes, M = 1 sectors, S_sector = 32 bits
Subsector size is 8 bits
Subsector overhead is 3 bits
Wasted subsector
A1
B11
C1
D1
A2
B12
C2
D2
A3
B21
C3
D3
A4
B22
C4
D4

Figure 4.9: Three possible configurations of the three data-layout parameters: total number of active probes, sector parallelism, and sector size. The figure shows a simplified MEMS-based storage device, where in (a) File A fits in one sector, whereas file B is split over two sectors, B_1 and B_2. The only sector of A in (a) is split into sectors A_1 and A_2 in (b) when doubling the number of probes per sector (the same goes for B_1 which is split into sectors B_{11} and B_{12}). The figure shows three configurations. The first configuration shown in (a) uses 2 out of 4 probes simultaneously, each accessing a 16-bit sector at a time. By using twice as many active probes as in (b), a probe accesses only half a 16-bit sector, so that 4 probes access two sectors in total simultaneously. In (c) the sector size doubles and one sector is striped across all probes. As a result, a probe accesses a quarter of a 32-bit sector. Increasing the sector parallelism as in (b) causes external fragmentation and thus seeks like from $B_{11}||B_{12}$ to $B_{21}||B_{22}$ ("||" means accessing in parallel), whereas increasing sector size as in (c) causes internal fragmentation, wasting capacity like A_3 and D_3.

the type of errors the device is prone to. We call the portion of user data of a physical sector, a logical sector. In disk drives, the ECC is one-tenth the size of the logical sector [71]. We assume that the size of the ECC overhead (S_{ECC}) is even larger in MEMS-based storage device, and is one eighth the size of the logical sector (S_{sector}), which is in agreement with figures available from the

IBM MEMS device. Thus, the ECC overhead is:

$$S_{\text{ECC}} = \left\lceil \frac{S_{\text{sector}}}{8} \right\rceil$$

Mechanical storage devices exhibit a physical overhead in order to address and access the user data. This physical overhead is a few bits that separate every two contiguous subsectors as Figure 4.9a shows. The separation bits (1) allow for data buffering before writing a subsector, and (2) keep the clock of the read channel running, so that the subsector can be fully read/written. Jacob *et al.* [64, Chapter 18, pages 650 – 652] provide an anatomy of the physical sector in disk drives. The physical overhead in disk drives has a small influence on the capacity, because it occurs once per sector. Conversely, in MEMS-based storage devices the physical overhead has to occur every subsector, because every probe accesses a subsector due to striping. We assume that the total physical overhead per subsector is three bits, which amounts to a period of 75 μs that is sufficient for processing.

From the above, striping a physical sector across K probes results in a physical subsector of size ($S_{\text{p-subsector}}$):

$$S_{\text{p-subsector}} = \left\lceil \frac{S_{\text{sector}} + S_{\text{ECC}}}{K} \right\rceil + 3. \tag{4.1}$$

To avoid very small capacities, we assume that a physical subsector is larger than or equal to 8 bits. To avoid seeks within an access to a subsector, the maximum physical-subsector size is smaller than the subtrack size (i.e., the portion of a track located in one probe field) $\frac{\text{field length}}{\text{bit length}} = \frac{100000}{40} = 2500$ bits.

4.4.3 Influence of Each Parameter

This section studies the influence of each data-layout parameter individually on the three design targets (i.e., energy, performance, and capacity). We simulate against the `ext3-4K` trace. Table 4.2 lists the additional settings of our simulated MEMS-based storage device based on the studies conducted in the previous two sections. The PSM of Figure 4.3 is deployed. These settings are complementary to those in Table 3.3 and Table 3.4 in Section 3.2. The complete methodology is detailed in Chapter 3 and Section 4.2.2.

Total Number of Active Probes (N)

Performance A MEMS-based storage device has a large number of probes to enhance performance. Increasing the number of probes a sector is striped across shortens the read/write time, because the subsector size decreases as Equation (4.1) shows. Figure 4.9a and Figure 4.9b show that doubling the number of active probes from 2 to 4 results in smaller subsectors a single probe has to access per sector; compare A to $A_1||A_2$ ("||" denotes parallel access). Thus,

Table 4.2: Additional settings of the MEMS model based on the conducted studies in Section 4.2 and Section 4.3. The settings are complementary to those in Tables 3.3 and 3.4

Parameter	Setting	Unit
Shutdown time-out	1	ms
Shutdown policy	Energy efficient	
Number of active probes	64 – 4096[a]	probes
Sector parallelism	1 – 16[a]	sector
Logical sector size	0.5 – 8[a]	KB

[a] We select values that are a power of two.

the time to read/write a striped sector is effectively the time a probe takes to read/write one subsector:

$$t_{\text{RW}-\text{subsector}} = \frac{S_{\text{p-subsector}}}{r_{\text{probe}}}, \tag{4.2}$$

where $S_{\text{p-subsector}}$ is the size of the physical subsector from Equation (4.1), and r_{probe} is the data rate per probe.

Simulating against `ext3-4K`, Figure 4.10 plots the response time as a function of the number of probes per sector of size 512 bytes. Because the minimum subsector size is 8 bits, the maximum number of probes per sector ($K = \frac{N}{M}$) is 512 probes. The response times are normalized to the response time when deploying 64 probes (83 ms). Figure 4.10 confirms the significant influence of the number of probes on the response time. When the number of probes doubles, the response time approximately halves.

Energy Unlike the response time, which decreases as the number of probes increases, energy to access data does not decrease because more probes are switched on at the same time. Actuation energy, however, decreases. Actuators are powered on to keep the media sled in position on the X direction and to move it along Y. Increasing the number of probes decreases the subsector size, and thus shortens the distance the sled travels along Y, and subsequently the time it is held on X (compare Figure 4.9a to Figure 4.9b). Consequently, the actuation energy decreases. The total read and write energy per physical sector can be written as follows:

$$\begin{aligned} E_{\text{RW}-\text{sector}} &= E_{\text{probes}} + E_{\text{actuation}} \\ &= K \cdot P_{\text{probe}} \cdot t_{\text{RW}-\text{subsector}} + P_{\text{actuation}} \cdot t_{\text{RW}-\text{subsector}}, \end{aligned} \tag{4.3}$$

where K is the number of probes per sector, P_{probe} is the power a probe dissipates to read or write one single bit, and $P_{\text{actuation}}$ is the power dissipated by

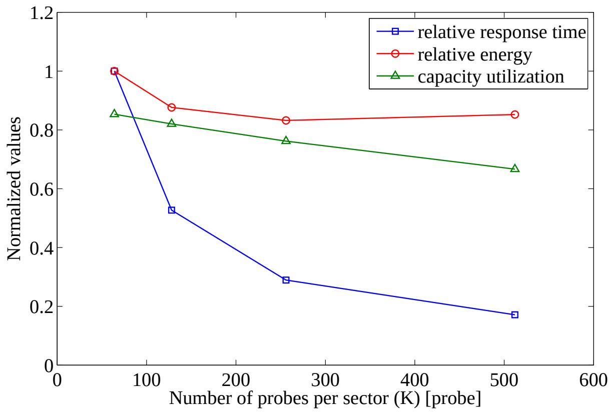


Figure 4.10: Relative average response time, relative total energy consumption, and capacity utilization of the simulated MEMS-based storage device as a function of the number of active probes deployed per sector. Simulation are carried out for sector parallelism (M) of one sector and a sector size (S_{sector}) of 512 bytes. Capacity is normalized to the (raw) physical capacity of the device.

both X and Y actuators. We derive an analytical model of the actuation energy in Section 3.2.3. Note that increasing the number of probes has no influence on the actuation energy, since the probes touch the medium during the actual read and write operations only. From Equation (4.3), we can observe that the reduction in the total read/write energy is bounded by the energy consumed by the probes to read or write, explained by Amdahl's law[1].

Figure 4.10 confirms this bound and shows the energy figures normalized to the energy when deploying 64 probes (13.6 J). As the number of probes increases, the actuation energy decreases, as the total energy does. The energy difference between every two successive points decreases as the number of probes increases, since the actuation energy becomes less prominent. A minimum point exists at 256 probes, after which energy starts increasing slowly. This increase is due to the additional overhead bits that need to be accessed (Equation (4.1)), which becomes more noticeable (compare three to four bits of overhead per sector in Figure 4.9a respectively Figure 4.9b). Figure 4.10 reveals that the number of probes has a larger influence on the response time than the energy consumption, since it influences the read/write time more than the read/write energy.

[1]Speeding up a part of proportion f of a system by a factor s results in an overall speedup of $\frac{1}{(1-f)+\frac{f}{s}}$. The $(1-f)$ term of the denominator tells that the overall speed up is limited by the rest of the system that cannot be enhanced.

Capacity Several physical bits are needed per subsector to enable its accessibility (Section 4.4.2). As the number of probes increases, the subsector size decreases and the relative overhead per sector increases. As a result, the (effective) capacity of the device decreases. Figure 4.10 shows the utilization of the physical capacity of the device (approximately 3 GB). The values of the capacity are normalized to the raw (physical) capacity of the device. Figure 4.10 shows a loss of 35% (approximately 1 GB) in capacity when deploying 512 probes due to the physical separation bits and the ECC data.

Further, Figure 4.10 shows that the three design targets compete when designing a MEMS-based storage device: a gain in performance results in a loss in capacity. Also, performance gain can compete with energy reduction.

Sector Parallelism (M)

Sector parallelism is the number of sectors that are simultaneously accessible from the storage medium. It deals with the number of probes a sector is striped across. If a MEMS-based storage device has N total active probes that access M sectors simultaneously, the number of probes per sector (K) is:

$$K = \frac{N}{M}. \tag{4.4}$$

Performance Increasing the sector parallelism (M) results in fewer probes per sector (K). As a consequence, the subsector size increases (Equation (4.1)). Increasing the sector parallelism has one positive influence and two negative influences on the performance of MEMS-based storage devices. The positive influence is that increasing the subsector size reduces the overhead (Figure 4.9b versus Figure 4.9a), and thus decreases the read/write time of the overhead bits. On the other hand, one negative influence is that increasing the subsector size increases the number of data bits a probe has to access, which increases the data read/write time. The second negative influence is underutilizing the sector parallelism, if the the request size is not a multiple of the number of simultaneously accessible sectors. If a request demands L sectors from a MEMS-based storage device, which is capable of accessing M sectors simultaneously, the response time for the request (t_{request}) is:

$$t_{\text{request}} = t_{\text{RW}} + t_{\text{seek}} \text{ and} \tag{4.5}$$

$$t_{\text{RW}} = \left\lceil \frac{L}{M} \right\rceil \cdot t_{\text{RW-subsector}}, \tag{4.6}$$

where $t_{\text{RW-subsector}}$ is the read/write time per subsector calculated as presented in Equation (4.2). For example, accessing file D in the MEMS-based storage device shown in Figure 4.9b incurs underutilization of those probes associated with D_2, because it has no useful data. Equation (4.5) shows that in addition to the read/write time, a seek time exists. The seek time includes the initial seek

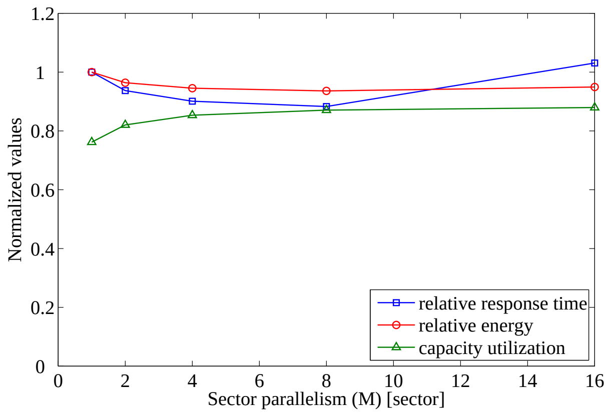


Figure 4.11: Relative average response time, relative total energy consumption, and capacity utilization of the simulated MEMS-based storage device as a function of the sector parallelism. Simulations are carried out for a total number of probes (N) of 256 probes and a sector size (S_{sector}) of 512 bytes.

time in addition to the seek times incurred due to accessing non-contiguous sectors (see Section 3.2.2 for details on the seek model).

Figure 4.11 shows the response times for various sizes of the sector parallelism normalized to the response time when sector parallelism is 1 (24 ms). It shows that sector parallelism of eight sectors exhibits the shortest response time when deploying 256 probes. Setting the sector parallelism larger than eight sectors under-utilizes the active probes, and results in longer response times. Thus, sector parallelism can be tuned based on the characteristics of the expected workload to diminish the two negative influences.

Energy A discussion similar to the performance applies to the energy consumption of MEMS-based storage devices as a function of sector parallelism. The total energy consumed to satisfy a request of L sectors is:

$$E_{\text{request}} = N \cdot P_{\text{probe}} \cdot t_{\text{RW}} + P_{\text{actuation}} \cdot t_{\text{RW}} + P_{\text{seek}} \cdot t_{\text{seek}} \,. \tag{4.7}$$

The first two terms are calculated as in Equations (3.7)–(3.10) in Section 3.2.3. In addition to the two negative influences on performance, a third negative influence on energy exists. As the parallelism increases, the subsector size increases, which extends the time the medium is held still on the X direction, and increases the traveled distance along Y. As a consequence, the actuation energy increases. Nonetheless, tuning the sector parallelism as done for the performance (in the face of the first two negative influences) and employing a larger number of probes simultaneously (in the face of the third influence) reduce the energy consumption. Figure 4.11 shows that, indeed, the energy consumption is minimal for sector parallelism of eight sectors. We deploy 256 probes simultaneously to minimize the third influence since it is the minimum in Figure 4.10. The values are normalized to the energy when sector parallelism is one sector (11.2 J).

Capacity Increasing the sector parallelism has a positive influence on the capacity, because the subsector size increases and thus the overhead per sector decreases. Figure 4.11 shows that the loss in capacity of about 0.3 GB made (see Figure 4.10) is earned back by formatting with sector parallelism of eight sectors. Better still, a further reduction in energy consumption and enhancement in performance is possible.

Sector Size ($S_{\mathbf{sector}}$)

Equation (4.1) shows that increasing the size of the logical sector increases the physical subsector size, which is also the result of increasing the sector parallelism. As the sector size increases, the subsector size increases too, resulting in the same influences when increasing the sector parallelism (Figure 4.9c). The main difference between increasing the sector parallelism and increasing the sector size is that the former can underutilize probes, if sectors are not requested. On the other hand, increasing the sector size can underutilize probes, if the sector does not fully contain useful user data. Our analysis shows the same trends to those in Figure 4.11.

Sector Parallelism versus Sector Size

Sector parallelism and sector size are two seemingly similar solutions to increase the size of the subsector to mitigate the imposed overhead per subsector. However, sector parallelism and sector size have different effects on the usage of the storage space, which in turn influences the performance and energy consumption. Increasing the sector parallelism increases external fragmentation, since related sectors are not necessarily spatially co-located. For example, accessing sectors B_{11}, B_{12}, B_{21}, and B_{22} shown in Figure 4.9b can not be done entirely in parallel, causing one more seek and a read or write of $B_{21}||B_{22}$ after $B_{11}||B_{12}$. On the other hand, increasing the sector size increases internal fragmentation, because sectors are not fully utilized, if the file system lacks intelligent placement techniques. For example, A_3 and A_4 in Figure 4.9c are wasted storage space.

External fragmentation increases seek and read/write operations, whereas internal fragmentation increases storage-space underutilization. Sector parallelism and sector size can be tuned based on the workload to enhance performance at yet large capacity.

4.4.4 Design Space

This section studies the design space of the data layout of MEMS-based storage devices composed of all feasible configurations of the three layout parameters discussed in Section 4.4.1. As Table 4.2 shows, we consider seven different settings of the number of probes, five settings of the sector parallelism, and also five settings of the sector size. All settings are a power of two, since the maximum number of probes and the sector size are power of two.

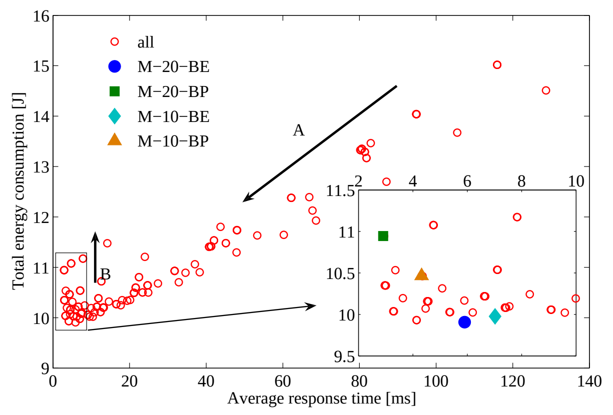
We present the three different views of a three-dimensional design space, where every configuration of the parameters (represented by a circle in Figures 4.12a–4.12c) exhibits a certain response time, energy usage, and capacity when simulating against the `ext3-4K` trace. In total there are 175 configurations out of which 20 configurations are infeasible, because they either exhibit a subsector size smaller than 8 bits (the minimum) or larger than 2500 bits (the maximum).

Figure 4.12a plots the energy consumption versus the response time. We can identify two trends referred to as trend A and trend B. Trend A shows that as the number of probes increases, the response time and energy consumption decrease. However, trend B shows that at a certain point the energy consumption increases as the number of probes increases, because the energy to access the overhead bits becomes noticeable.

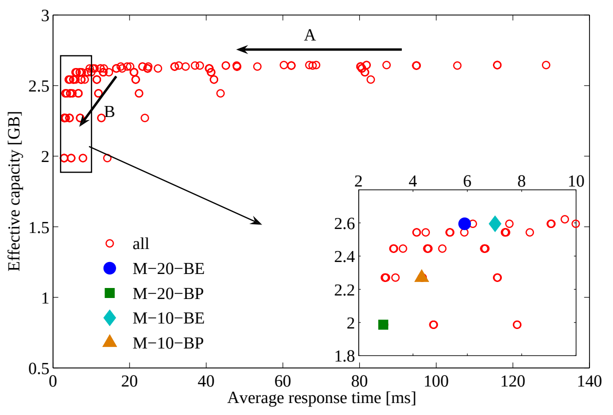
Figure 4.12b shows the effective capacity versus response time. Trend A shows that increasing the number of probes reduces the response time while retaining most of the device physical capacity. This trend corresponds to sector parallelism larger than one sector and/or sector size larger than 512 bytes as shown in Figure 4.11. Unlike trend A, trend B shows that a loss in capacity occurs, if the sector parallelism is one sector and/or sector size is 512 bytes as shown in Figure 4.10. By deploying large sector parallelism and/or sector size, we can retain a large part of the physical capacity at a negligible loss in response time as shown by the points around 2.5 GB.

Figure 4.12c shows the effective capacity versus the energy consumption. One trend similar to the previous figure can be observed, namely trend A. Trend B shows that a loss in capacity is accompanied by a loss in energy for configurations with a large number of probes. The reason is that employing many probes simultaneously increases the overhead per sector, causing a loss in energy as well as capacity, unlike trend B in Figure 4.12b. Although increasing the overhead increases the response time, a larger decrease in response time occurs by decreasing the number of data bits per probe (see Figure 4.10), which results in an overall decrease in response time.

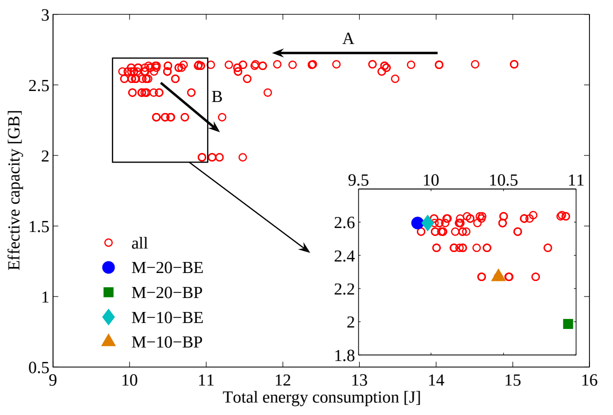
Zooming in on the parts where the optima could be found in Figures 4.12a–4.12c, we find that no overall optimal solution exists, but a set of Pareto optimal points. Thus, trade-offs are inevitable. We plot the best-energy (`M-20-BE` and `M-10-BE`) and best-performance (`M-20-BP` and `M-10-BP`) configurations when deploying 4096 respectively 2048 probes. Here, "M" denotes to MEMS,



(a) Energy consumption versus response time



(b) Effective capacity versus response time



(c) Effective capacity versus energy consumption

Figure 4.12: Trade-offs between the three targets (energy consumption, performance, and capacity) for all 155 feasible configurations when simulating against the `ext3-4K` trace

20 corresponds to the nominal throughput of 20 MB/s, and “BP” denotes to best performance. Note that the configurations with `M-20-*` are Pareto optimal[2], whereas the others are not. We explain the choice of these particular configurations in Chapter 6.

A discussion similar to that for the `ext3-4K` trace goes for other traces. Workloads of different properties (i.e., request size and address alignment) have different optimal performance, energy, and capacity configurations. An overall optimal configuration does not exist, so that a trade-off is inevitable. In Chapter 6, we revisit the configuration of MEMS-based storage devices under different workloads, and compare to Flash memory.

4.5 Summary

This section devises three policies that enhance the energy-efficiency, timing performance, and the capacity of MEMS-based storage devices. We analyze the characteristics of MEMS-based storage devices, and subsequently devise the Power State Machine (PSM), which consists of five operation modes (seek, active, idle, shutdown, and inactive) and has no startup state, unlike HDDs.

We show that a MEMS-based storage device consumes approximately 40% of its total energy consumption when idle, so that power management is necessary. Simulation results show that a fixed-timeout power management policy achieves (near) optimal energy saving (95% of the the idle energy). Also, we show that avoiding an immediate shutdown reduces the response time by 10%.

To allow for aggressive shutdowns, a MEMS-based storage device must reduce on the shutdown energy. We propose to exploit the unique structure to reduce the shutdown energy by using the potential energy of the springs only to move the sled toward the center position. Our simulations show a reduction by up to 10% in total energy compared to that when actuators are fully used during shutdown.

The third policy is the data-layout policy. We formulate the sector striping problem in MEMS-based storage devices and propose to exploit knowledge of the expected workload to configure the data layout. Simulation results show that such exploitation enhances the energy-efficiency and the timing performance by approximately 10%, while increasing the capacity utilization by 10% relative to the case if no knowledge is exploited. Our study shows that for some configurations trade-offs between response time, energy consumption, and capacity are necessary.

[2]A Pareto optimal solution dominates other solutions on at least one account. In our work, a Pareto optimal configuration outperforms other configurations either in terms of response time, energy consumption, or capacity.

CHAPTER 5

Wear-Leveling Policies

MEMS-based storage technology faces the wear challenge. A MEMS-based storage device is susceptible to various types of wear, of which probe wear is the most significant. Probe expiry results in a field fault that typically spans thousands of sectors. Maintaining an even level of wear across all probes prevents premature expiry of probes. Wear leveling provides a fully functional MEMS-based storage device, and potentially extends its lifetime.

In this chapter, we present three policies to keep the probe wear balanced during the lifetime of the device. The policies provide different trade-offs of device lifetime, the timing performance and energy-efficiency of a MEMS-based storage device. We devise the optimal policy that maximizes the lifetime of the device.

5.1 Wear in MEMS-Based Storage

In this section, we discuss the different types of wear, the cause of the wear, and the effects that wear has on the behavior of MEMS-based storage devices.

5.1.1 Types and Causes of Wear

MEMS-based storage devices have two main types of wear: (1) probe wear and (2) medium wear [72, 73, 74]. Note that MEMS-based storage devices have no rubbing surfaces, so that wear of bearing does not exits, unlike in disk drives. Probe wear inhibits the tip of the probe to write or read nanometer-sharp bits.

Medium wear inhibits the individual location on the medium to store or to retain a bit for a certain amount of time. Medium wear affects the device on a sector basis, whereas probe wear affects a probe field, which spans thousands of sectors.

The cause of probe wear varies depending on the recording technology. Usually, wear is caused by friction at high load on the probe tip, high temperature, and high velocity. In addition, Bhushan *et al.* [73] indicate that tribochemical reactions can take place. Likewise, the medium wears due to several factors including high temperature, and contact with the probe.

In thermomechanical recording like in the IBM MEMS-based storage device, a probe tip wears due to heating and probe–medium contact. By pushing a heated probe tip against a polymer medium a nanometer-scale indentation is created in the medium to represent a bit of value of “1”. By writing bits repeatedly a probe looses its sharpness leading to a complete wear of the probe tip.

In phase-change and ferroelectric recording, where the probe tip has to maintain good conductance, wear is even more challenging than in thermomechanical recording. In these techniques a probe tip wears due to high temperature and high velocity. Bhushan *et al.* [73] give an extensive study of the causes of wear in these two techniques. Probes and medium might be lubricated to reduce probe wear.

Based on the literature [72, 73, 74] and discussions with physicists, we assume the following in our work:

- The write operation is the main cause of probe wear and medium wear. Writes incur mechanical forces in thermomechanical recording, or heat the tip to high temperatures in phase-change recording. Therefore, we use *the number of written bits per probe as a metric of wear*. That is, the larger the number of written bits, the more significant the wear. We assume the effects of reading on probe wear to be small enough to be negligible.
- A probe wears a few orders of magnitude faster than an individual bit location on the medium. This is because a probe surface is in constant contact during read/write, whereas bit locations on the medium are in contact with a probe just very shortly.
- A probe can write *at least* 100 times the capacity of its storage field (i.e., 10^9 bits in our model) before it starts to function unreliably. The rating of 100 represents the minimal requirement based on promising experiments [29, 75].

Physicists are enhancing the endurance of the probes and the medium. For example, they research various types of coating materials for the probes and the medium. Also, various materials are explored to produce probes for better endurance [75]. However, the research in this area is beyond the scope of this dissertation. In the remainder of this chapter, we focus on probe wear.

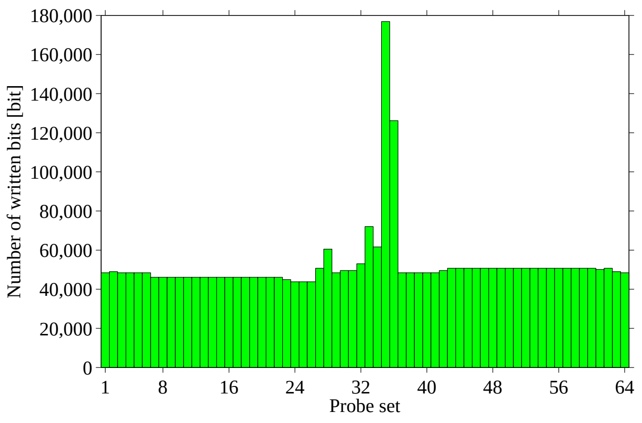


Figure 5.1: Distribution of wear across probe sets when exercising our simulated MEMS-based storage device with the `scenarios` trace. Note that the peaks span several noncontiguous areas of the address space.

5.1.2 Uneven Wear and its Effects

A sector is striped across a probe set to elevate the data rate of a MEMS-based storage device (Section 4.4). Since all probes in a probe set write exactly the same number of bits, probes within a probe set wear evenly. Wear leveling is applied on the probe set level.

Individual probe sets can write a different number of bits depending on the workload, the file system, and the mapping from the logical block address (LBA) to the physical block address (PBA). That is, the wear phenomenon at the probe level manifests itself in a problem of *uneven wear* of probe sets at the device level.

Some probe sets can wear much faster than others. Figure 5.1 shows the uneven use of probe sets in our simulated MEMS-based storage device when exercised with the `scenarios` trace (Section 3.3). Uneven wear of probe sets influences a MEMS-based storage device from the following perspectives:

Reliability If some probe sets wear out before others, their respective storage fields become inaccessible. As a result, user data located in these fields are lost.

Performance If some probe sets wear before others, the number of probes that can operate in parallel decreases. As a result, the data transfer rate of the device decreases.

Energy Wear of probe sets reduces the probe parallelism and increases energy consumption: reducing probe parallelism reduces the number of bits accessed in parallel. Consequently, the sled moves longer distances along Y and stays still for a longer time along X, increasing the actuation energy (Section 4.4.3).

Capacity A loss of just one probe results in a loss of its storage field which is typically several megabytes in capacity. A loss of a probe set reduces the device capacity by several hundreds of megabytes.

5.1.3 Device Life

Our objective is to preclude premature expiry of probes in a MEMS-based storage device in order to prevent the effects of uneven wear (Section 5.1.2). Rephrasing, our objective is *maximizing the lifetime of the individual probes, so that they live throughout the entire lifetime of the device,* and thus expire more or less simultaneously. The **device lifetime** is the total (aggregated) number of bits written by all probes of the device before it expires.

Let us assume an example MEMS-based storage device of 10 probes, each can write a maximum number of 10^9 bits before it expires. Figure 5.2 shows the two extreme cases of the life of this device. The best-case life happens when each probe lives the entire lifetime (i.e., 10×10^9 bits) of the device (the green dashed line). The worst-case life happens when probes expire one after another (the red dotted line). The best-case life and the worst-case life demonstrate how the maximum lifetime of the device can be achieved with a different lifetime of the individual probes. Therefore, probe lifetime is the main concern. Our objective life is the best-case one. An example of the life of a device with no wear leveling is shown by the blue curve.

Suppose that, in practice, a device is assumed expired if 10% of the probes expire. Figure 5.2 shows that the device writes 1×10^9, 7×10^9, 10×10^9 bits in the worst-case, typical, and best-case life, respectively, before it expires. That is, wear leveling increases the number of bits written before the device expires, namely the device lifetime. Observe that if the threshold is 100% the device lifetime is achieved by any (even no) policy, so that it is not a concern. At 100%, the only difference between wear-leveling policies is the probe lifetime to prevent the four effects discussed in Section 5.1.2.

Summarizing, wear leveling must maximizes the lifetime of the individual probes to preclude the effects on the reliability, timing performance, energy-efficiency, and the capacity of a MEMS-based storage device. In addition, the device lifetime as a fifth target is also equally important. This is because in practice a device is abandoned, if admitting a write request would expire one of the probe sets. In other words, the practical threshold is in fact 0% (and not 100% or even 10%), since user data must not be lost.

To maximize the lifetime of the individual probes, we should minimize the consumption of the probe write cycles, which can be achieved by wear leveling. A wear-leveling policy maintains simultaneous growth of wear across all probes. Graphically, a wear-leveling policy should make the blue curve as

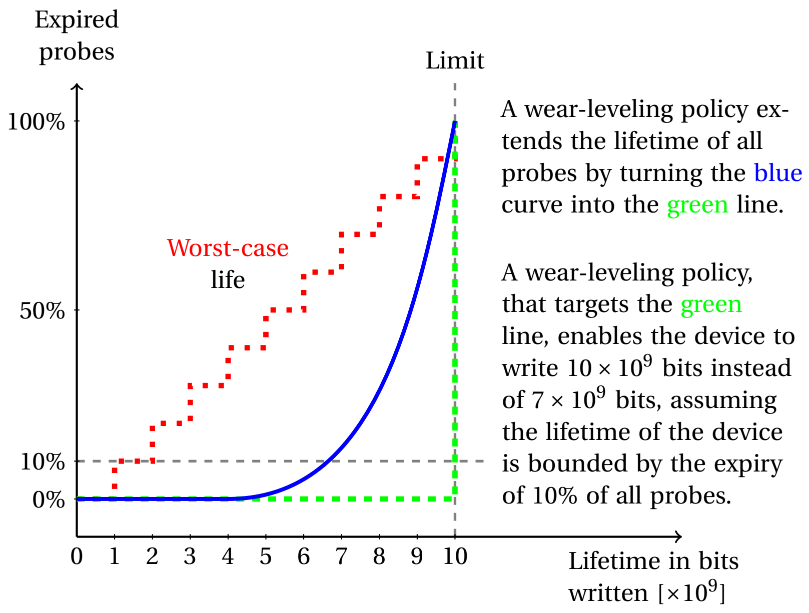


Figure 5.2: A graphical illustration shows the best-case, worst-case, and typical life of a MEMS-based storage device.

steeply as possible, and stretch the blue curve toward the limit. Ideally, a policy should turn the blue curve into the green dashed line.

Wear leveling enables us to utilize the device fully for a certain period of time; this period can be one order of magnitude longer compared to that if no policy is deployed as Figure 5.1 suggests. When all probe sets approach the end of their lifetime, the user can be advised to migrate his data to a new device and the write operation of the device can be blocked, for example, to prevent any data loss.

In the model of our simulated MEMS-based storage device, we assume that all probes have the same endurance of 10^9 bits. In practice, endurance varies due to several factors including manufacturing. However, this variation is much smaller than the variation due to uneven use. This assumption does not affect our work on wear leveling, since probes are still used unevenly. Thus, wear leveling is still needed to preclude premature expiry.

Our wear-leveling techniques can be combined with other techniques to extend the limit (the dashed vertical line in Figure 5.2) on the maximum number of bits written. One technique to extend the limit is to read a sector before writing it. Doing so, the device writes only those bits that have been changed, and thus reduces the effective number of bits written per sector, virtually ex-

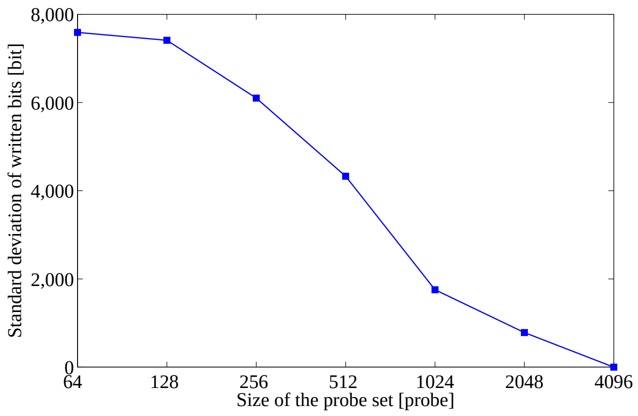


Figure 5.3: Wear is inherently leveled across probes as the size of the probe set increases. The figure is for the `scenarios` trace when no wear-leveling is deployed.

tending the limit.

Like the limit, the lifetime bar can also be raised. We can deploy k-out-of-n information dispersion techniques, such as the Information Dispersal Algorithm [76]. These techniques encode a sector into n different pieces and map them to n different probes (of a probe set). With these techniques, a sector is still recoverable as long as at least k of its assigned probes are still operational to retrieve k pieces, raising the bar in Figure 5.2 above 0%. Similar techniques are deployed in the IBM MEMS device [29] and Redundant Array of Inexpensive (or Independent) Disks (RAID) organizations, such as RAID-5.

5.1.4 Uneven Wear and Parallelism

Equation (4.4) shows that the size of the probe set and the number of probe sets are related. Increasing the size of the probe set, reduces the number of probe sets, and vice versa. Increasing the size of the probe set levels the wear distribution across probe sets, since the number of probes, that write the same number of bits, increases. We use the standard deviation of the number of bits written per probe set as a metric of imbalance (see Section 5.3.2). The smaller the standard deviation the more evenly the wear is distributed across probe sets.

Figure 5.3 plots the standard deviation of written bits across probe sets versus the size of the probe set for our simulated MEMS-based storage device; a 64×64 probe array. The figure shows the decaying trend as the size of the probe set increases. If the probe set encompasses all the probes of a MEMS-based storage device, across which a sector is equally striped, then each probe writes the same number of bits. The resulting distribution of written bits is a constant distribution, whose standard deviation is zero as shown in Figure 5.3 for the size of 4096 probes.

From above, increasing the size of the probe set is beneficial for wear leveling. It is, however, constrained by several factors including the cost and the permitted power budget for operating the device. A mobile storage device can be housed in various packages of different power budgets. For instance, the CompactFlash (CF) package has a power budget of 3.3 W, whereas SecureDigital (SD) has a 0.3 W power budget. The CF package can afford to power a few thousand probes, whereas the SD package allows a few hundred only. Also, increasing the probe-set size reduces the effective capacity of the device, because more housekeeping bits, such as control data, are needed as explained in Section 4.4.2.

Summarizing, the importance of wear leveling in MEMS-based storage devices increases when targeting small storage packages of limited power budget.

5.2 Causes of Uneven Wear

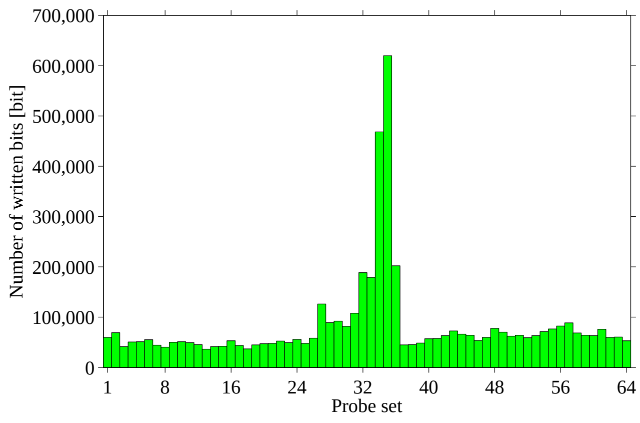
Uneven wear of probe sets in a MEMS-based storage device is caused by unevenly distributed accesses to areas on the storage medium by I/O requests. An I/O request r is represented as a tuple: $(t_\mathrm{r}, A_\mathrm{r}, S_\mathrm{r}, O_\mathrm{r})$, where t_r is the arrival time of the request, A_r is the logical address of the starting block, S_r is the size of the request, and O_r is the operation of the request: read or write.

The properties of a request r that affect the mapping to the physical space are the address (A_r) and the size (S_r). The address determines the starting probe set. The size determines the consecutive probe sets as well as the load on each set per request. These properties determine the wear of the probe set, if the operation (O_r) is a write operation. We quantify the influence of request address and size on uneven wear by testing two hypotheses using our traces (Section 3.3). The hypotheses are:

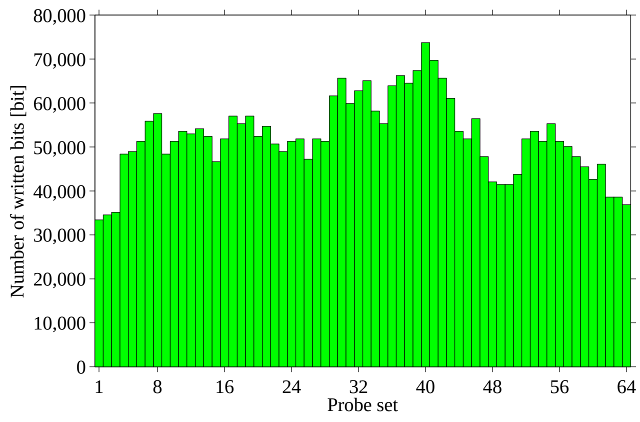
Hypothesis 1 : An uneven distribution of the number of requests across probe sets causes uneven wear.

Hypothesis 2 : An uneven distribution of the size of requests across probe sets causes uneven wear.

We use the captured traces to test our hypotheses as follows. For hypothesis 1, we resize all requests in a trace to the average request size. We leave the address unchanged, thus any uneven wear of probe sets observed is attributed to the request address (A_r) only, and not to the request size. We take the average request size to quantify the influence of the request address versus the request size on uneven wear. For hypothesis 2, we map all requests in a round-robin fashion across all probe sets to distribute requests evenly over



(a) Wear distribution due to number of requests



(b) Wear distribution due to size of requests

Figure 5.4: Wear distribution across probe sets when simulating against the `scenarios` trace. The figures confirm Hypotheses 1 and 2, that the number of requests and their size cause uneven wear. We observe that some probe sets wear significantly faster than other sets. The influence of the number of requests on uneven wear is larger than the request size.

probe sets. We keep the size of individual requests unchanged, so that any uneven wear observed is attributed to the request size (S_r) only.

Figure 5.4a and Figure 5.4b show the distribution of written bits for both hypotheses, respectively, for a full run of the `scenarios` trace. The figures confirm the deviation from the ideal equal distribution of writes. We observe that wear can be larger by an order of magnitude for some probe sets than others; for example, compare probe set 35 to 16 in Figure 5.4a. From the figures, we observe the larger influence of the number of requests compared to the request size on uneven wear. Retrospectively, this observation is a result of the fact that the request size is bounded in practice, whereas the number of requests is not.

Note that each probe set is responsible for several noncontiguous physical areas on the medium, because of the logical data layout depicted in Figure 3.3b on page 42. As a result, the peak in Figure 5.4a is not due to just one hot area, but actually several noncontiguous hot areas [1].

In addition, we test the hypotheses against the `multimedia` and `iozone` traces, and arrive at conclusions in conformity to those for the `scenarios` trace. Thus, we confirm the influence of the number of requests and the size of the request with emphasis on the larger influence of the former compared to the latter. Based on both hypotheses, we can construct a policy that optimally levels the wear. The policy should write an equal number of sectors to each probe set, while cycling through the probe sets in a round-robin fashion. We call this policy the sector-based round-robin policy (or briefly `rrSector`). We detail it in Section 5.4.

5.3 Methodology

In the following, we devise three wear-leveling policies and evaluate them. The difference between these policies boils down to two choices: (1) the selection of a request for remapping (**candidate request**), and (2) the selection of a probe set to remap the candidate request to it (**victim probe set**). These selections affect the probe lifetime and thus the device lifetime, resulting in a different lifetime of the device with each policy.

Remapping the process of finding an available PBA other than the static PBA decided by the mapping explained in Section 3.2.4. In all policies, if a victim probe set cannot be found, the default set is selected. The **default victim probe set** is determined according to the LBA to PBA mapping presented in Section 3.2.4.

We discuss all policies for our simulated MEMS-based storage device. A sector is striped across 256 probes (i.e., the size of a probe set). Given that we have 4096 probes in total, our MEMS-based storage device has $\frac{4096}{256} = 16$ probe sets. We discuss the effect of different sizes of the probe set in Section 5.7.4. We present our study of the policies for the `scenarios` trace and report for the other traces in Section 7.5. We use the same MEMS model we have used throughout the dissertation (Tables 3.3, 3.4, and 4.2).

For all policies, we preserve the X and Y offsets of a sector in its default storage field when remapping to another storage field as shown in Figure 5.5. As a result, seeks due to unavailability in storage space are not included. But

[1] We investigated identifying requests of hot sectors, and remapped them to the coldest probe set in an attempt to level wear. We concluded that hot data as well as cold data wear probe sets unevenly. Hot data are frequently accessed, so that their respective probe sets are heated. But also cold data arrive in large amounts that map to the same sets, so that they heat probe sets too.

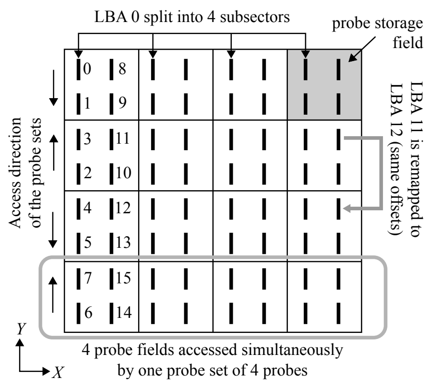

LBA 0 split into 4 subsectors
probe storage field
LBA 11 is remapped to LBA 12 (same offsets)
Access direction of the probe sets
0
1
3
2
4
5
7
6
8
9
11
10
12
13
15
14
4 probe fields accessed simultaneously by one probe set of 4 probes
Y
X

Figure 5.5: An illustration on a simple MEMS-based storage device shows that when remapping to another probe set, we select the sector with the same physical *X* and *Y* offsets, such as remapping LBA 11 to LBA 12.

seeks to reposition the medium due to the distance it moved during remapping are included. Although seeks due to space unavailability are likely to be incurred in practice, excluding them allows us to single out the direct influences of remapping on the response time and energy-efficiency of MEMS-based storage devices. The direct influences are: (1) processing needed for remapping, (2) lowering the sequentiality, and (3) remapping of subsequent read requests.

5.3.1 MEMS Translation Layer

We envisage MEMS-based storage devices having a MEMS Translation Layer (MTL) at the the interface between the device and the physical driver. The design of the MTL can borrow many design concepts from the design of the Flash Translation Layer (FTL). In practice, the MTL is a part of the firmware that runs on the micro-controller of the storage device as shown in Figure 5.6.

The MTL requires persistent storage to keep track of the LBA-to-PBA mapping, because of wear leveling. The map can be stored on a dedicated memory, such as Ferroelectric RAM (FeRAM), as shown in Figure 5.6. Alternatively, like in some FTL designs [77], the infrequently updated part of the map can be stored on the MEMS storage module itself to reduce the cost for additional memory. In any design, the map has to be loaded into a fast memory at run time to speed up search operations.

The MTL enforces the wear-leveling policy by dynamic remapping. The lifetime of a probe is represented in terms of its write cycles (i.e., the number

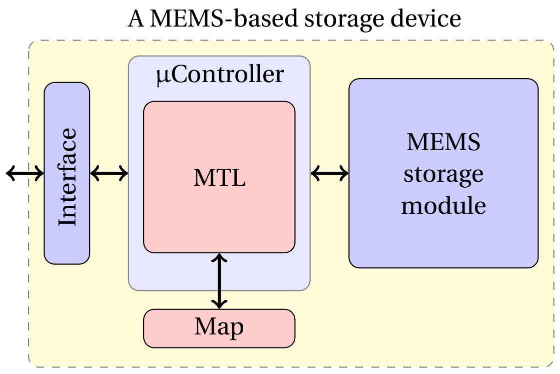


Figure 5.6: An illustrative block diagram of a MEMS-based storage device shows that the MEMS Translation Layer (MTL) can be implemented in the firmware that runs on the μController.

of bits a probe writes before it wears out). The MTL needs to track the wear by maintaining a write counter for written bits per probe. The counters are stored in a persistent place and used by the wear-leveling policy to decide for the appropriate probe set at the remapping time.

For our experiments, we implemented an MTL and coupled it with the DiskSim simulator. The MTL receives I/O requests, processes them, and then forwards them to DiskSim for servicing. Implementations of the wear-leveling policies took place in the MTL. The MTL applies the wear-leveling policy, monitors wear of probes and hotness of data, and remaps logical block addresses to physical block addresses. It also collects statistics about probe wear and the LBA-to-PBA mapping for our research.

5.3.2 Standard Deviation

The green dashed line in Figure 5.2 corresponds to an equal distribution of wear across all probe sets, where all probe sets write the same number of bits. The equal distribution of wear across probe sets is represented by a constant probability distribution of the written bits, where one single value occurs 100% of the time. The constant probability distribution has a standard deviation of zero. Uneven wear causes a deviation from the constant distribution, so that the standard deviation becomes larger than zero.

As the green dashed line indicates in Figure 5.2, our objective is to maximize the lifetime of the individual probes. Toward this end, a wear-leveling policy must maintain simultaneous growth of wear across probe sets. To measure the imbalance in wear distribution across probe sets, we use the standard deviation as a metric, and pursue its evolution over time to monitor the simultaneous growth of wear.

Note that the policies, offered in this work, strive to maintain the wear across all probe sets at more or less the same level all the time; that is they maintain a constant probability distribution. As a result, the policies prevent *by construction* the creation of tailed probability distributions. This prevention is achieved in all policies by (1) triggering the remapping upon every write request, and (2) cycling through the probe sets. The policies compete to transform the resulting semi-uniform probability distribution of written bits across probe sets into an ideal constant probability distribution. Therefore, the standard deviation is a sufficient metric for the decrease in the width of the semi-uniform probability distribution to reach the constant probability distribution.

As mentioned in Section 5.1.3, a wear-leveling policy increases the lifetime of a MEMS-based storage device. We analytically quantify the lifetime of the device for each of the three policies we present next. The quantification is presented in terms of the utilization (u) of the total write cycles of all probes: the ratio of the effective maximum number of bits that can be written when adopting a certain policy to the theoretical maximum. The effective maximum number of bits is the number of bits that can be written to the device, so that admitting one write request more would expire one of the probe sets (graphically it is the point at which the blue curve in Figure 5.2 leaves the horizontal line of 0%). The theoretical maximum is $M \cdot K \cdot 10^9$, where M is the total number probe sets and K is the number of probes per set.

5.4 The Sector-Based Round-Robin Policy

The sector-based round-robin policy levels wear across probe sets by writing every arriving sector to a subsequent probe set in a round-robin fashion.

Selection of the candidate request: `rrSector` considers each arriving I/O request as a candidate, and remaps its sectors.

Selection of the victim probe set: `rrSector` cycles in a round-robin fashion through probe sets. It maintains an index for cycling.

Figure 5.7 illustrates how `rrSector` works. It considers requests to the device as a sequence of sectors. Every sector is mapped to a subsequent probe set in a round-robin fashion, regardless of the request it belongs to. Consequently, sequential sectors of the same request are mapped to different probe sets, thus the sequentiality of requests is not preserved. For example, request A in Figure 5.7 is mapped to probe set 1, 2, and 3.

The round-robin mapping scheme of `rrSector` guarantees simultaneous growth of wear across all probes. It further guarantees that the difference in wear between any two probe sets is limited to the subsector size ($S_{\text{subsector}}$) at any time instance. Figure 5.8 shows that the standard deviation is maintained

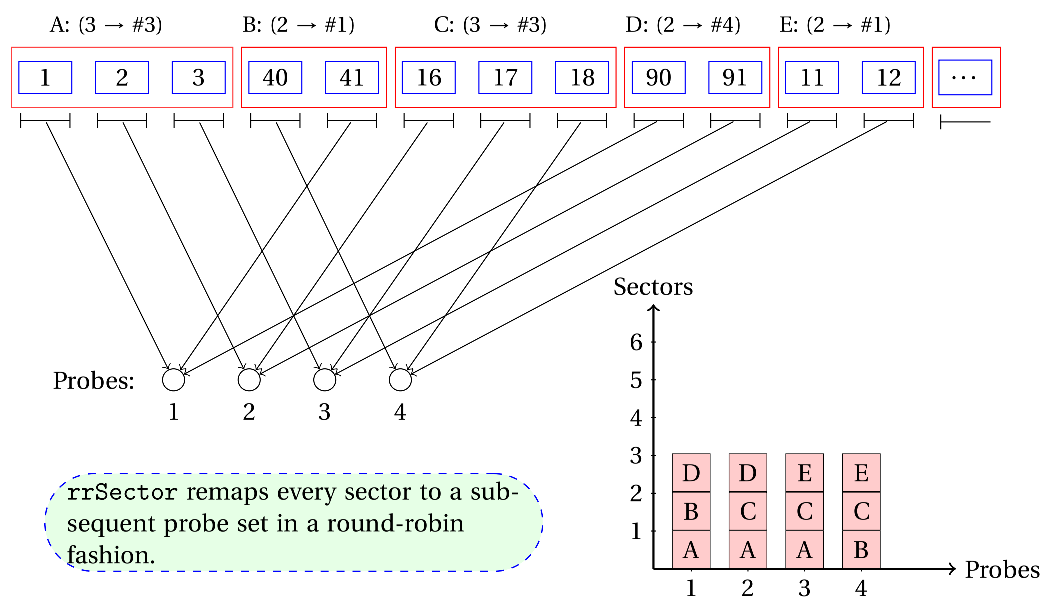


Figure 5.7: Remapping sectors to probe sets using the sector-based round-robin policy (`rrSector`). B: (2 → #1) means that request B writes two sectors to its default probe number 1. For simplicity, we talk in terms of probes and sectors; in reality, however, we talk in terms of probe sets and subsectors, due to striping.

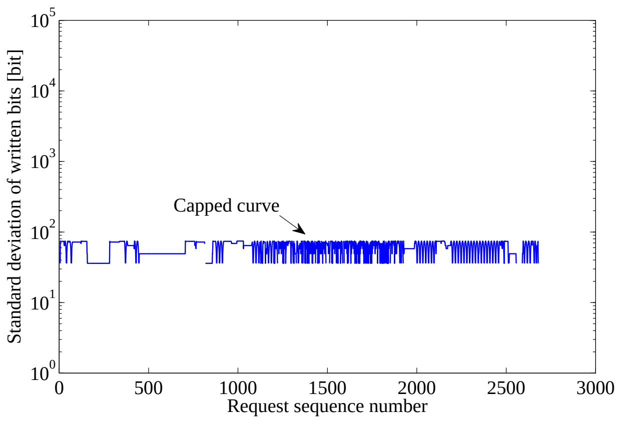


Figure 5.8: Standard deviation of written bits across the probe sets for the `scenarios` trace when employing the sector-based round-robin policy

at a certain level and does not rise above it. In fact, the standard deviation is capped in Figure 5.8, because of the bounded difference. The policy achieves an equal distribution of wear across probe sets every full round-robin cycle, reducing the standard deviation to zero at several points such as the 800-th request. These points are shown as a discontinuity, since zero corresponds to minus infinity on a logarithmic scale.

The `rrSector` policy maximizes the utilization of probes, because it reduces the difference in wear to the minimum, namely the subsector size. The total number of unwritten bits is bounded by $M \cdot K \cdot S_{\text{subsector}}$, where M is the number of probe sets and K is number of probes per set. The utilization for the `rrSector` policy becomes $u = 1 - \frac{S_{\text{subsector}}}{10^9}$.

Optimality The difference in wear between any two probe sets in practice cannot be smaller than the subsector size. This is because the subsector size is the smallest atomic unit a probe (in a probe set) writes per sector. As a result, the `rrSector` policy represents the optimal wear-leveling policy that can be realized in practice, since it bounds the difference by the subsector size. That is, `rrSector` maximizes the lifetime of the individual probes and the device beyond what other policies can do.

The maximization of the probes and the device lifetime comes at a cost however. That is, because `rrSector` remaps the sectors of a request to non-contiguous locations, the policy, compromises on the timing performance and energy-efficiency. This reduces the applicability of the `rrSector` in practice, particularly for mobile battery-powered devices. In the following, we present alternative policies that favor timing performance and energy-efficiency. We take `rrSector` as a reference point with respect to lifetime, since it results in the optimal device lifetime.

5.5 The Coldest-Probe Policy

This section presents a policy that preserves the request in its entirety in order to minimize the influence on the performance and energy-efficiency, while extending the probes and the device lifetime. For the sake of brevity, we represent the degree of wear (i.e., the number of written bits) by temperature. That is, the more worn a probe, the higher its temperature, and vice versa.

Selection of the candidate request: `coldestProbe` marks a request arriving to the MEMS-based storage device as a candidate, if its default probe set is hotter (more worn) than at least one other probe set.

Selection of the victim probe set: `coldestProbe` remaps candidate requests to the coldest probe set at the arrival time of the request. The policy keeps track of the number of written bits of each probe set.

Figure 5.9 demonstrates that even though request C maps to probe 3 by default, the policy remaps it to probe 2. Probe 3 is the most worn probes, whereas probe 2 is the least worn probe at the arrival of request C. The policy assumes that remapping a request to the coldest probe set contributes to minimizing the variance in wear across probe sets. The efficacy of the `coldestProbe` policy depends on the size of the arriving request. That is, a large request size increases the imbalance, whereas a small one has a little influence. We observed this in the difference of the maintained levels of the standard deviation for the three traces, since they have different dynamics.

Figure 5.10 shows two peaks in the standard deviation at the 1300-th and 2200-th request. The peaks are approximately one order of magnitude larger than the maintained level of the standard deviation. Similar peaks were observed with the other two traces, particularly for the `iozone` trace such peaks are two orders of magnitude larger than the maintained level. Analyzing the mapping, we found that these peaks are caused by a flurry of large requests that always map unevenly to two or three consecutive probe sets. These flurries write not only with cold probe sets, but also with hot probe sets too, which increases the standard deviation. Worse, the cold probe set remains relatively cold to its hot neighbors, so that successive large requests to the same probe sets still map the same way, further increasing the deviation. Assume that a request arrives to the MEMS-based storage device in Figure 5.5 that writes to LBA 1 through LBA 3. After this request, probe set 0 (the first row of probes) has written one sector, whereas probe set 1 has written 2 sectors. Suppose the next request writes to LBA 9 through LBA 11. The request still maps to probe sets 1 and 2, because probe set 1 is still relatively the coldest. After the second request, probe set 2 has written 4 sectors compared to just 2 sectors for

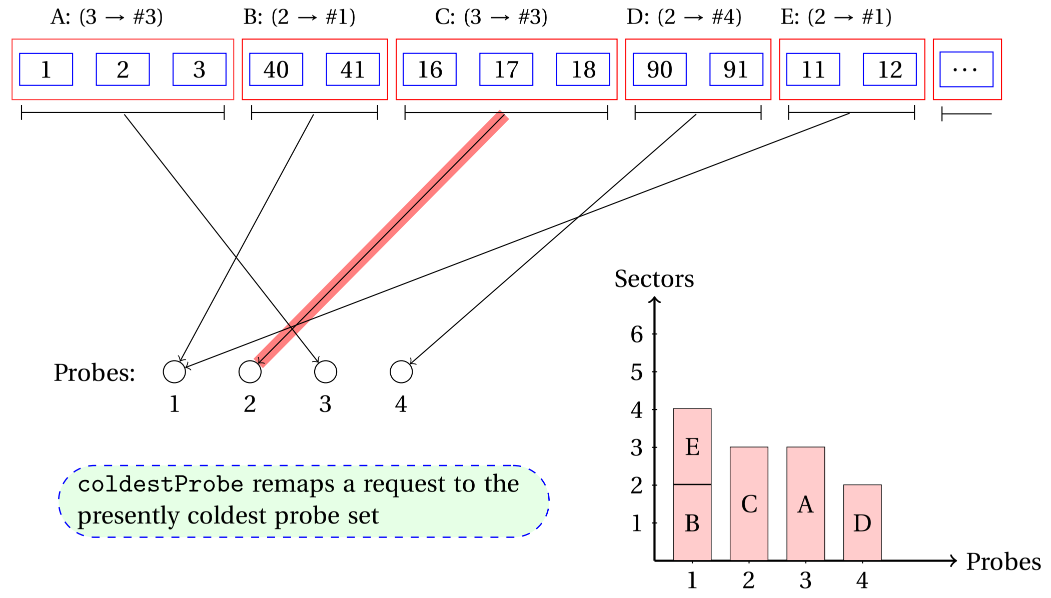


Figure 5.9: Remapping sectors to probe sets using the coldest-probe policy (`coldestProbe`). Remapped request are marked in thick edges.

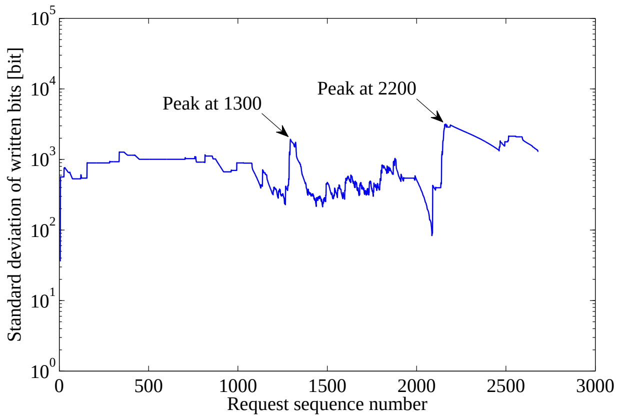


Figure 5.10: Standard deviation of written bits across the probe sets for the `scenarios` trace when employing the coldest-probe policy. Peaks exist due to large sequential requests.

probe sets 1. Observe that the difference in wear between the two probe sets has raised from 1 sector to 2 sectors, which results in an increasing standard deviation. Small requests arrive later, which map entirely to cold sets, leveling the wear again.

Such flurries can be potentially harmful for the device lifetime, since they can expire some probe sets before others; recall that we want to avoid the red dotted line in Figure 5.2. On the other hand, the flurries certainly harm the performance and energy-efficiency of MEMS-based storage devices. If a probe set gets overly heated by a flurry of streaming requests, for example, `coldestProbe` remaps all future requests, whose default probe set is the heated one. This remapping is likely to happen for a reasonably large number of requests, since the flurry is large. For the ensuing remappings, a MEMS-based storage device has to lookup an alternative storage location and seek to it, incurring performance and energy costs [2]. The costs are likely to be incurred for a large number of requests, hurting the response time and energy-efficiency of the device. The incurred costs can be particularly large, if they are caused by streaming (large) requests and affecting best-effort (small) requests.

We can solve the overheating issue (i.e., the peaks at 1300 and 2200) by remapping the request, so that not only the first probe set is cold, but also all the other probe sets are cold too. However, the consequence is that the suc-

[2]This scenario resembles an inefficient usage of a RAID-0, Redundant Array of Inexpensive (or Independent) Disks, organization. Instead of dispatching requests to all available disks in order to elevate the throughput and reduce seeks, requests are dispatched to one disk at a time.

cessiveness of probe sets is not preserved, and thus the contiguity of the respective physical sectors is not maintained. This influences the response time and energy-efficiency of MEMS-based storage devices. Therefore, we prefer to keep the sequentiality preserved.

Because this policy cannot deal with such flurries, it does not necessarily maintain simultaneous growth of wear and can lead to large underutilization. The utilization for `coldestProbe` is $u \approx 1 - \frac{M-1}{M} \cdot \frac{1}{L_{\max}}$, where $L_{\max}$ is maximum number of sectors per request and M is the number of probe sets. Observe that as M decreases the utilization increases, reaching $u \approx 1$ when all probes make up one probe set ($M = 1$). That is, reducing the number of probes sets increases the utilization. In fact, this is in conformity to our experimental results. In Section 5.7.4, we show that for large set sizes such peaks disappear, so that the coldest-probe policy remains a viable design choice. Next, we present a variant of the coldest-probe policy that handles flurries in the workload and controls the influence on the performance and energy-efficiency.

5.6 The Barrier-Based Policy

The barrier-based policy (`barrier`) is inspired by parallel computing [78]. A barrier, in parallel computing, is a synchronization technique that halts a process at a certain point in its execution from proceeding until all other processes reach the point. In the `barrier` policy, the barrier represents the number of written bits.

Our goal is to maintain simultaneous growth of wear across probe sets, so that, optimally, they reach their maximum lifetime simultaneously. Toward that ultimate goal, we set incremental subgoals for the probe sets, so that all probe sets reach each subgoal simultaneously. These subgoals make up our barriers down the operation time of the probe sets. The idea is that lining up probe sets at every barrier avoids racing between them toward the ultimate goal or barrier.

Selection of the candidate request: The barrier-based policy maps a request to its default probe set unless the probe set has already crossed the barrier. Then, the request becomes a candidate one and gets remapped. Figure 5.11 exemplifies such a case for request C, which is remapped to probe 2, because probe 3 has already crossed the barrier of two sectors due to request A. The policy maps a request with large sequentiality to its default probe sets as long as one of them has not crossed the barrier.

Selection of the victim probe set: The barrier-based policy selects for a candidate request a sufficiently cold probe set. That is, it chooses a probe set, so that if the request is remapped to this probe set, the probe set reaches or crosses the barrier with the minimum distance (i.e., number of written bits). Ideally, this distance should be zero. Remapping with the minimum distance

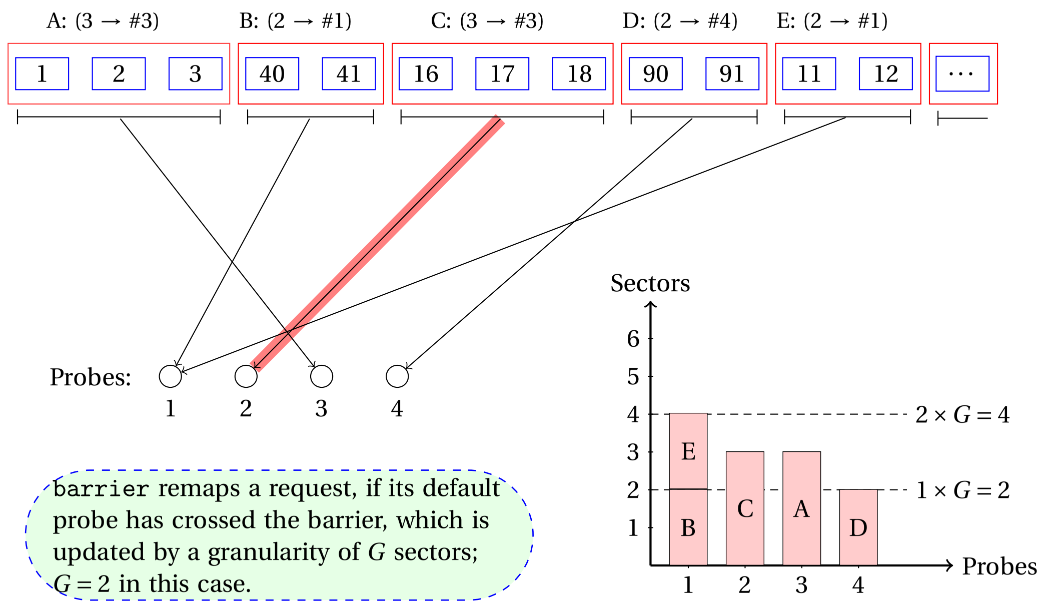

A: (3 → #3)
B: (2 → #1)
C: (3 → #3)
D: (2 → #4)
E: (2 → #1)
1
2
3
40
41
16
17
18
90
91
11
12
· · ·
Probes:
1
2
3
4
barrier remaps a request, if its default probe has crossed the barrier, which is updated by a granularity of G sectors; G = 2 in this case.
Sectors
6
5
4
3
2
1
E
B
C
A
D
2 × G = 4
1 × G = 2
Probes

Figure 5.11: Remapping sectors to probe sets using the barrier-based policy (`coldestProbe`). Remapped request are marked in thick edges.

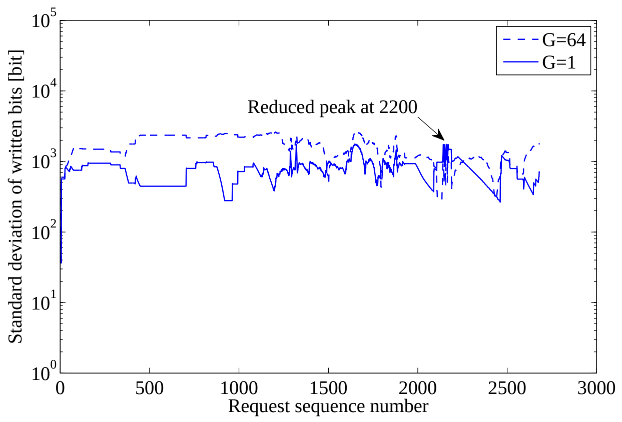


Figure 5.12: Standard deviation of written bits across the probe sets for the `scenarios` trace when employing the barrier-based policy with $G = 1$, and 64 sectors. Increasing G increases the local deviation from leveled wear.

allows the barrier-based policy to minimize the difference between probe sets. If no victim probe set is found, the default probe set is selected.

Updating the barrier: If all probe sets have crossed the current barrier (or subgoal), the barrier is incremented to the next multiple of a granularity G that is a parameter of the policy. In Figure 5.11, the barrier is incremented to $2 \times G$ after request D, since all probes reached or crossed the previous barrier: $1 \times G$. The granularity G represents a trade-off between uneven wear and the number of remappings. Setting the granularity G is particularly important, since very small G leaves no room for remappings, and large G defers remapping, resulting in either case in bad wear leveling. Our experiments suggest that the granularity should be larger than or equal to one sector and smaller than twice the maximum request size. The barrier can be updated with a dynamically adjusted step as our experiment results suggest in Section 5.7.4.

Figure 5.12 confirms the policy's capability of dealing with the causes of uneven wear distribution. The peak at 1300 shown in Figure 5.10 has disappeared, thanks to the barrier. The second at 2200 still appears but at a smaller magnitude. The figure also shows that uneven wear increases as the barrier granularity increases.

Again, we assume that the device reaches its end, if writing would lead to the expiry of one of the probe sets. The total number of unwritten bits is bounded by $M \cdot K \cdot \lceil \frac{L_{\text{max}} \cdot S_{\text{subsector}}}{G} \rceil \cdot G$, where L_{max} is the maximum number of sectors per request. The utilization for `barrier` becomes $u \approx 1 - \frac{L_{\text{max}} \cdot S_{\text{subsector}}}{10^9}$.

Similar to the `scenarios` trace, we tested the three policies against the `multimedia` and the `iozone` traces. The results agree with those obtained for the scenarios trace. In the following, we compare these policies from different perspectives for the three traces.

5.7 Comparison

The design of a wear-leveling policy involves trade-offs: a policy that is effective at wear leveling can be resource demanding and/or can influence the response time and energy-efficiency of MEMS-based storage devices. In this section, we compare the three wear-leveling policies (i.e., rrSector, coldestProbe, and barrier) with respect to the lifetime, the response time and energy-efficiency, and the storage requirement of our simulated MEMS-based storage device. We study three settings of the barrier in the barrier-based policy with $G = 1, 8$, and 32 sectors to extract possible trends. For comparison, we also show the figures when deploying no wear-leveling policy (called noop). We present the results for the scenarios trace and discuss for the other traces.

5.7.1 Device Lifetime

In the study of the policies, we analytically quantified the device lifetime. This section evaluate the device lifetime with each of the three policies by tracking the proliferation of the wear across the probe set with the maximum wear. The maximum wear is an indication of the reduction in the device lifetime (in bits written) as the most worn probe set expires. Graphically, we are measuring the point at which the blue curve in Figure 5.2 leaves the 0% threshold.

The number of bits as a metric of lifetime is generic across different usage patterns. This is because various usage patterns write different number of bits per time unit, resulting in different lifetime of the device. At design time, the designer translates the number of bits into days or years depending on the expected application. For example, assume an application that sends 100 requests to the device on average per day, and the device expiry limit is 50,000 bits. Taking the scenarios trace as an example of a usage pattern, the 50,000 limit intersects with the curve of noop at the 1500-th request and intersects with the barrier at the 2200-th request in Figure 5.13. This boils down to a lifetime of 15 respectively 22 days, increasing the lifetime by a factor of 1.46.

Figure 5.13 plots the development of the maximum number of bits of the most worn probe set for the scenarios trace. The figure illustrates the benefit of implementing wear leveling. That is, any of the three policies results in an increased lifetime compared to the noop policy. Further, the coldestProbe and barrier polices are relatively much closer to the optimal rrSector policy than to the noop policy.

As expected, the rrSector policy reduces the wear of the most worn set. It results in the minimum number of bits written per probe set in practice. Figure 5.13 shows that the most worn set exhibits less wear with rrSector compared to the other two policies. Observe that the difference to the other

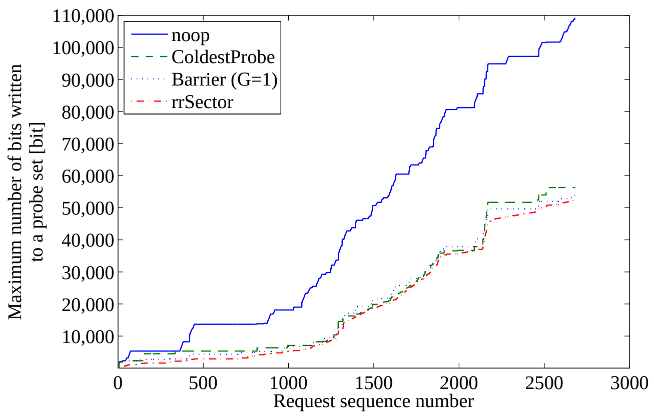


Figure 5.13: Evolution of the maximum number of bits written per probe set for the `scenarios` trace. The figure has scale granularity of 10,000 bits, which amounts to 625 sectors.

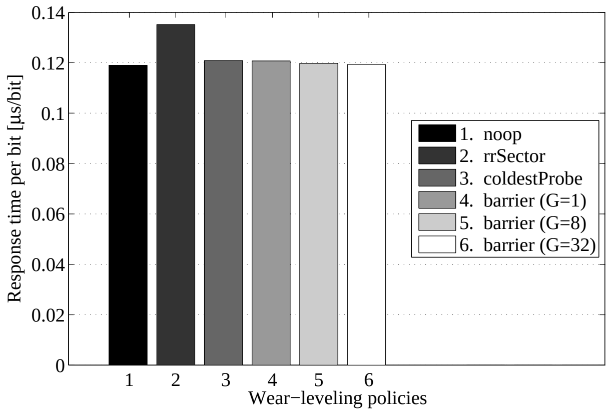
policies is relatively large, since the figure has scale granularity of 10,000 bits, which amounts to approximately 625 written sectors; recall that a probe writes only $\frac{4096}{256} = 16$ bits per sector, so that $\frac{10000}{16} = 625$ sectors.

Figure 5.13 shows that `barrier` achieves better leveling than the `coldestProbe`, since it can handle flurries of large requests. The figure reveals that `barrier` exhibits occasionally larger wear than `coldestProbe`. This is because barrier maps a request to its default probe set as long as the probe set has not crossed the barrier, leading to temporary overheating. On the other hand, `coldestProbe` maps always to the coldest, and, therefore, avoids overheating due to small requests.

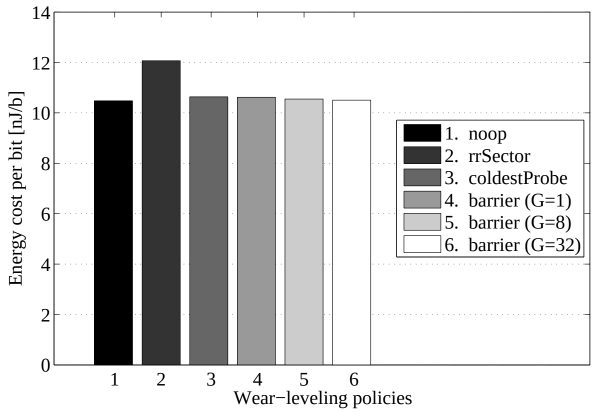
5.7.2 Timing Performance and Energy-Efficiency

In this section, we study the influence of each wear-leveling policy on the timing performance and the energy consumption of our simulated MEMS-based storage device. Recall from Section 5.3 that we remap a sector such that its X and Y locations in the new storage field are exactly the same as in the default field. As a result, seeks due to unavailability in storage space are not included. However, seeks to reposition the medium due to the distance it moved while remapping are included. We also study the influence of remapping of write requests on the subsequent read requests.

Figure 5.14a shows that the `rrSector` policy exhibits longer response time than the other policies. The policy incurs (1) larger request processing overhead, and (2) reduce the sequentiality of requests more than the other policies,



(a) Response time



(b) Energy consumption

Figure 5.14: Per-bit response time and energy consumption of our simulated MEMS-based storage device for the wear-leveling policies when simulating against the `scenarios` trace.

since it remaps on a sector basis. As a consequence, the response time per bit of the MEMS-based storage device increases by about 20% for the `rrSector` policy compared to `noop`.

The influence of the other two policies is smaller than that of the `rrSector`, since they preserve the sequentiality. Figure 5.14a shows that the policies other than the `rrSector` one exhibit the same performance. In practice, however, the difference in favor on `barrier` is more pronounced, since additional seeks are incurred due to space unavailability when remapping.

Similar to performance, Figure 5.14b shows that the energy consumption of a MEMS-based storage device increases as the sequentiality decreases. Further, an additional indirect influence on energy exists: since remapping causes

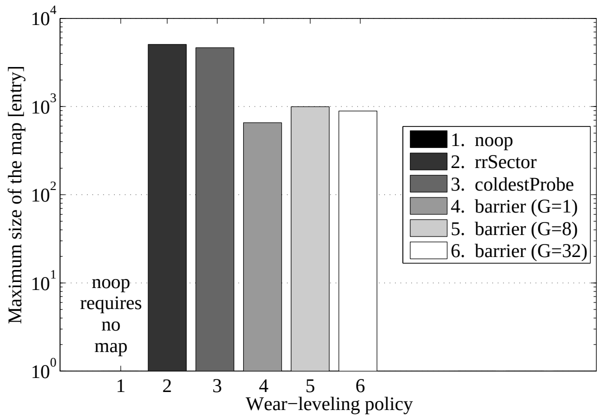


Figure 5.15: The size of the LBA-to-PBA map required by each policy for a full run against the `scenarios` trace

more mechanical activities, the media sled spends more time in seeks and turnarounds. Consequently, a MEMS-based storage device has less opportunity to shut down for energy saving.

5.7.3 Storage Requirement

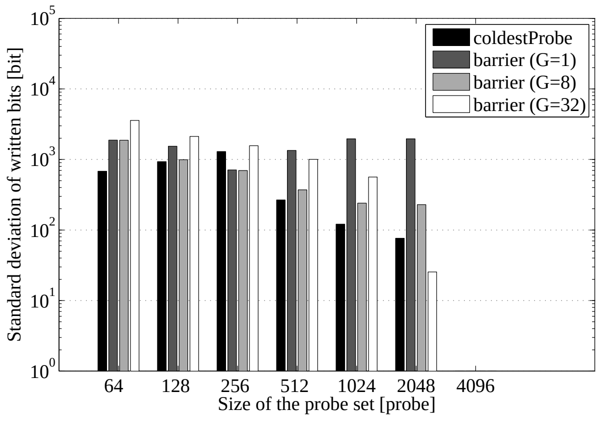
The dynamic remapping from the LBA to the PBA required for wear leveling demands a permanent store to record the mappings, called the map. The mappings are used by the device in the subsequent requests to read or overwrite the respective LBAs.

Figure 5.15 shows the maximum size of the map for a full run of every wear-leveling policy against the `scenarios` trace. Across all traces, the round-robin policy exhibits the largest map size, since it remaps every LBA (or sector). The `coldestProbe` policy exhibits almost the same size as the round-robin policy do, since it remaps virtually every request to the coldest probe set. Since the barrier policy remaps only, if a probe set has crossed the barrier, it demands less storage.

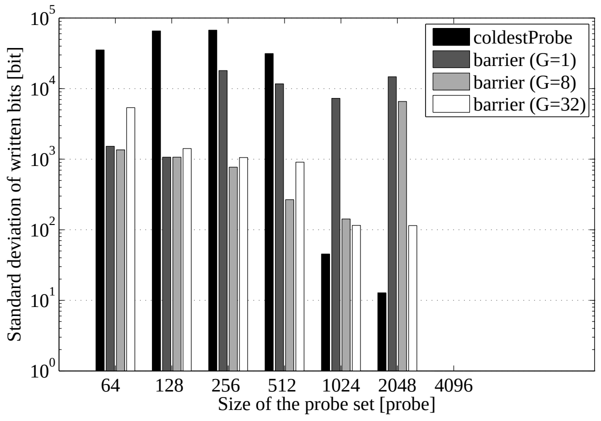
Figure 5.15 reveals no fixed relation between the map size and the granularity. Although one would expect that as the granularity increases, the map size decreases, since fewer remappings occur. However, the interplay with the request size obscures this relation.

5.7.4 Wear Leveling and Parallelism

The previous part of this work fixed the size of the probe set to 256 in order to account for a possible influence. We repeated the previous studies for different



(a) The `scenarios` trace



(b) The `iozone` trace

Figure 5.16: Standard deviation of written bits as a function of the size of the probe set. For traces that exhibit flurries of large requests, such as the `iozone`, the `coldestProbe` policy performs worse than the `barrier` policy for small probe sets. In general, as the probe-set size increases, the standard deviation declines faster with `coldestProbe` than `barrier`.

sizes of the probe set, starting from 64 up to 4096 probes with an increasing factor of power of two. This section discusses our findings.

In agreement with the trend shown in Figure 5.3, Figure 5.16a and Figure 5.16b show that, in general, when the size of the probe set increases, the standard deviation decreases. The figure shows the standard deviation of written bits across probe sets after a full run of the `scenarios` trace.

Figure 5.16a and Figure 5.16b show that the `coldestProbe` can perform better or worse than the `barrier` policy for small probe-set size depending on the workload. With workloads that exhibit frequent flurries of large requests, such as the `iozone` trace, `coldestProbe` performs worse (Figure 5.16b). On the other hand, `coldestProbe` performs better for workloads with few flurries (Figure 5.16a), such as the `scenarios` trace.

Figure 5.16b shows that the `coldestProbe` policy outperforms the `barrier` policy when the probe-set size is larger than or equal to 1024 probes. Our investigations reveal that the peaks (shown in Figure 5.10) disappear for large probe sets. In Section 5.5, we deduced that these peaks are due to large requests that span two to three consecutive probe sets. For large probe-set sizes, requests span only one probe set, so that the peaks disappear.

When the probe-set size increases, the number of bits a probe writes per sector decreases. As a result, the granularity of the `barrier` policy becomes relatively high, so that leveling occurs late. This explains the worse performance of `barrier` when the probe-set size increases. The `barrier` limitation could be fixed by deploying a granularity that is a fraction of sector size. But the `coldestProbe` policy becomes then more attractive, because of its simpler implementation.

Figure 5.16a and Figure 5.16b show that wear leveling of one variation of the `barrier` policy changes relative to the other variations from one trace to another. This change suggests that updating the barrier by a predetermined static step does not always result in the best leveling. In other words, updating the barrier should be dynamic to pursue the request-size dynamic of a workload. One way to update the barrier is to monitor the size of the incoming requests and adjust the barrier based on the next level of wear desired. For example, the policy can remap large requests only, while small requests map to their default set. Such tuning amortizes and reduces the performance and energy costs. That is, the barrier is a mechanism that allows the `barrier` policy to control the trade-off between probes and device lifetime, on one side, and the performance and energy-efficiency on the other side.

As mentioned earlier setting the probe set size to the total number of available probes guarantees a fair distribution of wear, since all probes write the same number of bits per sector. As a result, the standard deviation becomes zero as shown in Figure 5.16a and 5.16b for the size of 4096 probes.

5.7.5 MEMS Design Trade-offs

The previous section demonstrates that a trade-off exists between the timing performance and energy-efficiency on one side, and lifetime on the other side. In addition, it demonstrates that the probe-set size influences the efficacy of the policies. We summarize our findings in Table 5.1. It ranks the policies from four design targets: (1) device lifetime, (2) response time, (3) energy consumption, (4) the LBA-to-PBA map size of a MEMS-based storage device, and (5) the implementation cost of the policy.

This chapter offered three policies that provide the designer with a continuum between maximum device lifetime, and maximum performance, and

Table 5.1: Ranking from 1 (best) to 4 (worst) of `rrSector`, `coldestProbe`, and `barrier` from different design perspectives

Wear-leveling policy	Device lifetime[a]		Response time	Energy usage	Map size	Implementation cost
probe-set size	< 1024	≥ 1024				
`rrSector`	1	1	4	4	4	2
`coldestProbe`	3	1	3	2	4	2
`barrier`	1	1	2	2	3	3
`noop`	4	4	1	1	1	1

[a] Unlike the other targets, the longer the lifetime, the better.

energy-efficiency of a MEMS-based storage device. Retrospectively, the continuum is achieved by triggering the remapping process at different granularity: a subsector (`rrSector`), a request (`coldestProbe`), or even a few requests (`barrier`). The `noop` policy never triggers remapping, hence deploys infinitely large granularity.

Table 5.1 summarizes the ranking of the policies from the perspective of the design targets. The ranking ranges from 1 (best) to 4 (worst). Two policies receive the same ranking when their influence on the respective target is comparable. A difference in ranking of n ($n \geq 1$) between two policies denotes to a significant difference, in the range of n orders of magnitude.

From a lifetime perspective, `rrSector` and `barrier` rank first, since they maximize the utilization and the lifetime of the individual probes with a little difference in favor of the former. For a probe set larger than 1024 probes, `coldestProbe` ranks first as well and outperforms `barrier`. But for a probe set smaller than 1024, `coldestProbe` gives no guarantee on even wear. Deploying no policy (`noop`) reduces the device lifetime and can lead to worst-case utilization (the red dotted line in Figure 5.2).

From performance and energy-efficiency perspectives, deploying no policy represents the best case, since no direct and indirect costs due to remapping are incurred. The second best policy is `barrier`, since remapping is triggered every few requests (i.e., the barrier). The second last is `coldestProbe`, which remaps every request, whereas `rrSector` ranks last which remaps every sector. Unlike seek time, seek energy is small relative to the total energy of a MEMS-based storage device (see Figure 4.2). As a result, `coldestProbe` and `barrier` ranks comparable.

For the map size, `rrSector` and `coldestProbe` rank comparable, since they (almost) remap every request. In contrast, `barrier` reduces the size of the map by an order of magnitude. The policy with the smallest map size, namely zero, is `noop` which never remaps.

Another dimension that we add to the ranking study, is the cost or the effort needed to implement each of the policies, which can be measured by the lines of code written or amount of time needed to come up with an appropriate implementation. This factor influences the amount of time to implement a certain policy, which in turn reflects in its cost. Here, `noop` incurs no implementation costs, whereas `rrSector` and `coldestProbe` incurs more cost to check the next probe in charge for remapping. The most expensive policy is `barrier`, which requires to implement mechanisms to update the barrier. Further, if the barrier requires dynamic updating, as suggested in Section 5.7.4, then even more implementation costs are incurred.

5.8 Wear-Leveling in Flash Memory

This section discusses the difference between probe wear leveling in MEMS-based storage and medium wear leveling in Flash memory. Tackling probe wear differs from tackling medium wear due to their difference in function. To ease the discussion, we use the analogy of temperature: hot data are frequently written data, hot (physical) sectors are worn sectors, and hot probes are worn probes. The temperature is a relative metric; for example a probe cools down if the temperature of the others increases relative to it and vice versa.

A probe (or the read/write head) is a means to read and write data, whereas the medium is a container that stores data. Henceforth, we discuss medium wear in terms of sector wear, since it is the basic unit of storage on the medium. It is essential to observe the difference in the cause of uneven wear between probes and sectors. Probe uneven wear is due to uneven number and size of writes dispatched to probes (Section 5.2). On the other hand, sector uneven wear is due to the coexistence of hot and cold data. For example, assume that we write a sector 10 times with one probe and we write ten other sector one time with another probe. Here, both probes have the same amount of wear, namely writing ten times the size of a sector. In contrast, the sectors have a different degree of wear; one is written ten times more than the others. This example demonstrates the essential difference in the cause of uneven wear of probes and sectors, which stems from the difference in their function.

The difference in function necessitates a different approach to cool down probes and sectors. That is, a hot probe cools down by avoid using it temporarily. On the contrary, a hot sector cools down by writing cold data in it. This is, because writing data in a sector makes it unavailable, and thus cannot be used anymore unless data are migrated out. On the other hand, a probe does not contain data, and is thus available all the time. A probe becomes unavailable, if its storage field is entirely filled. In our case, where the data layout fills the fields simultaneously (Section 3.2.4 on page 39), it is most probable that when a probe field is filled then the rest of the fields are filled too. At that point, the device is about filled. Thus, medium wear leveling must monitor the temperature of data for wear leveling, whereas probe wear leveling has no such obligation.

The difference in function has a third consequence. Recall that a sector is a container, so that it can be used only if it is available (i.e., contains no useful data). A probe can be used almost all the time, since it is merely a loader and not a container. Data should be migrated from a sector to make it available, so that the sector can be used to store new data for wear leveling. In other words, data migration is an intrinsic part of medium wear leveling to guarantee the circulation of hot data through all sectors in order to level the wear. Data migration causes wear itself since it incurs writing, therefore it should be minimized. Worse yet, cold data make up the majority of data in real-world workloads, so that minimizing the migration of a huge amount of data is essentially necessary for performance reasons. In contrast to medium wear leveling, probe wear leveling requires data migration in very special cases. We construct in the following a scenario to demonstrate that just peculiar workloads necessitate data migration in probe wear leveling.

Consider a workload that fills all probe storage fields, so that their respective probes reach the same temperature. After that, the workload erases the contents of half of the probe fields and then refills half of each of the cleaned fields. At that point, half of the probe fields are fully occupied, and the other half are half occupied. However, the probes of the fully filled fields are colder than those of the partially filled, and thus should be used for wear leveling. Any incoming write request after this points should be diverted to cold probes, but they are not available. To make them available, the device migrates data from cold-probe fields to hot-probe fields. Note that migration itself increases the wear of the hot probes, since it incurs writing. Therefore, migration is only worthwhile if the new data, that will be written to the cold-probe fields, are written more than once. This workload is considered peculiar, because it deletes selectively several noncontiguous areas accessed by certain probes.

Data migration in Flash memory has been the subject of a large number of works [77, 79]. Several techniques have been proposed to limit data migration, and thus to reduce its overheads. Because Flash writes at a page and erase at a block granularity (of several pages), data migration techniques have to provide compaction functionality to reduce unnecessary wear of blocks; for example by copying several cold pages from different blocks into one block. Probe wear leveling has no such task.

In summary, probe wear leveling is essentially simpler than medium wear leveling. A probe wear leveling policy does not need to establish the temperature of data, it must avoid writing with hot probes, and it rarely involves data migration. On the contrary, a medium wear leveling policy must establish the temperature of data for proper mapping, must write cold data to hot sectors for cooling down, and must migrate data to keep the circulation of hot data through sectors flowing.

5.9 Summary

This chapter addresses the problem of uneven wear of probes in MEMS-based storage devices. The wear on the probe level manifests itself as probe uneven wear on the device level. Uneven wear has serious consequences for the reliability, timing performance, energy-efficiency, capacity, and lifetime of MEMS-based storage devices. Therefore, wear leveling is a must.

We devised three wear-leveling policies that result in a different influence on the lifetime, timing performance, energy-efficiency of a MEMS-based storage device. Also, the policies differ in their respective implementation cost. One of the policies represents the optimal policy in practice that maximizes the lifetime.

We evaluate the policies by simulating against real-world traces. We find in a case study that wear leveling increases the device lifetime by a factor of 1.46. Our simulation results show that designing a wear-leveling policy involves trade-offs between the device lifetime, the required storage resource, and the resultant performance and energy-efficiency.

We present the sector-based round-robin policy, which achieves the best wear leveling, resulting in the maximum device lifetime. The policy maximizes the device the lifetime. However, because it breaks requests into sectors and remaps every sector, this policy increases the response time and the energy consumption by approximately 20% compared to the other two policies.

The second policy is called the coldest-probe policy, which remaps a request in its entirety to the probe with the least wear at the remapping time. This policy results in better performance and energy-efficiency than the previous policy for a comparable lifetime for large probe sets. In contrast, for small probe sets, the policy cannot cope with flurries of large requests, and can thus reduce the lifetime significantly.

The barrier-based policy sets barriers down the operation time of the device, where a barrier is a certain number of bits. The policy remaps requests in their entirety, so that all probe sets reach a barrier simultaneously. The policy has a smaller influence on the performance and energy-efficiency than the second policy. It incurs less remapping, and therefore requires less storage.

The barrier policy is capable of coping with flurries of large requests, and thus ranks second in lifetime after the optimal policy. The coldest-probe policy remains, however, a viable design choice for large probe sets, since it incurs less remapping overhead compared to the barrier policy. Unlike the other two policies, the barrier policy requires more implementation effort, adding to the cost of the device.

CHAPTER 6

Capacity-Modest Applications

In the previous two chapters, we have provided a set of policies that increase the viability of MEMS-based storage devices as a storage solution. To put the resulting MEMS-based storage device into perspective, we compare it to Flash memory in this chapter. The comparison highlights the capabilities of MEMS-based storage technology, positions it with respect to other technologies, and pinpoints potential issues of enhancement. We compare the two in terms of response time and energy consumption.

In the following, we compare our simulated MEMS-based storage device with a Flash card for capacity-modest applications, such as mobile phones and PDAs. These applications require relatively small Flash, so that the Flash cost is acceptable. We compare for two mobile environments: mixed-media and streaming. The mixed-media environment has a mix of best-effort and streaming applications. Streaming applications correspond to playing back or recording audio and video.

6.1 Methodology

The modeled MEMS-based storage device we use throughout this chapter has the Power State Machine (PSM) depicted in Figure 4.3 with the timeout set to 1 ms and 0 ms for mixed-media and streaming environments, respectively. The device employs the energy-efficient policy to shutdown (Section 4.3). We

Parts of this chapter have been presented at the 8-th ACM & IEEE International Conference on Embedded Software (EMSOFT'08), Atlanta, Georgia, USA [Khatib: 3].

employ the `coldestProbe` wear-leveling policy from Section 5.5. Recall that this is an economic policy that extends the lifetime of MEMS-based storage devices, while not significantly compromising on the timing performance and energy-efficiency. Further, the policy supports configurations with large number of probes, such as 1024 probes.

Table 3.4 lists the settings of our modeled MEMS-based storage device and Table 4.2 lists the settings of its Power State Machine (PSM). We scale the bit dimensions in our MEMS model to 40 nm × 40 nm, so that the formatted MEMS-based storage device has a capacity that is approximately equal to that of our Flash card: about 2 GB (Section 3.1.1). The scaling maintains a fair comparison, since seeks in the MEMS-based storage device span the whole physical dimensions of a probe storage field and thus the address space. Consequently, we report the worst-case for seek time and seek energy. Note that the per-probe data rate is preserved when scaling the bit dimensions, so that the read/write time is not influenced.

We compare with Flash memory, since it is well known for its short access time and energy efficiency. We choose the CompactFlash form, because it has superior performance to smaller forms like the Multimedia Card (MMC) and the Secure Digital (SD) card. Further, we do not choose high-performance cards like CF Extreme-III, because these cards pack more Flash chips at higher cost, and we use just a single-chip MEMS-based storage device. That is, we try to be as fair as possible to MEMS-based storage devices in terms of performance and cost.

6.2 Mixed-Media Environments

In mixed-media environments best-effort applications are intertwined with streaming applications. The share of each type of application depends mainly on the usage pattern of the mobile system. We compare for this environment by simulating against the `scenarios` trace, which has seven streaming sessions and nine best-effort sessions.

6.2.1 Configurations of the Data Layout

Designing a MEMS-based storage device constitutes a multi-objective optimization problem (Section 4.4.4). The designer has to trade off between the design targets: timing performance, energy-efficiency, capacity, and lifetime (Chapter 4 and Chapter 5). In this section, we select several configurations of MEMS-based storage and compare them to Flash memory. The configurations have different settings of three data layout parameters: number of probes, sector parallelism, and sector size.

We explore the design space of configurations as shown in Section 4.4.4 and select the overall-best configuration in terms of response time and energy

Table 6.1: Best configurations of our simulated MEMS-based storage device from a response-time, energy-consumption, and capacity perspective for the `scenarios` workload with the configuration `ext3-4KB`. Each configuration is a tuple (number of probes, sector parallelism, sector size). Here, “M” denotes to MEMS, 20 corresponds to the nominal throughput of 20 MB/s, and “BP” denotes to best performance.

Configuration	`ext3-4KB`		
M-20-BP[a]	(4096,	1,	4)
M-20-BE[b]	(4096,	16,	4)
M-20-BC[c]	(64,	4,	8)
M-10-BP[d]	(2048,	1,	4)
M-10-BE	(2048,	16,	2)

[a] M-20-BP: Overall-best performance configuration out of the configurations that have a nominal throughput of 20 MB/s.

[b] M-20-BE: Overall-best energy configuration

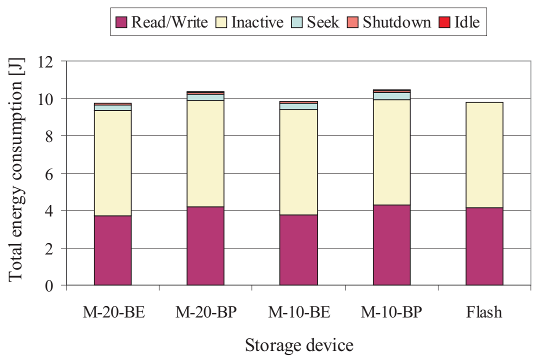
[c] M-20-BC: Overall-best capacity configuration

[d] M-10-BP: Best performance configuration out of the configurations that have a nominal throughput of 10 MB/s.

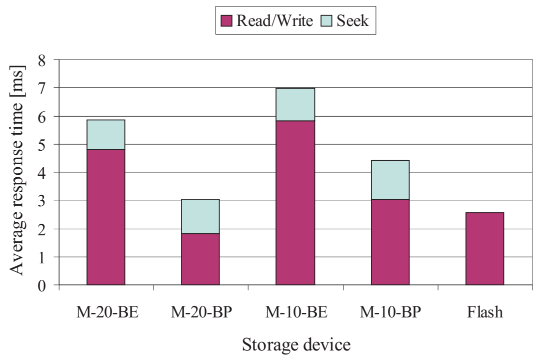
consumption, called M-20-BP (best-performance configuration) and M-20-BE (best-energy configuration), respectively. The letter M is for MEMS and 20 denotes the nominal throughput 4096×40 Kb/s = 20 MB/s. These configurations are highlighted in Figures 4.12a–4.12c. We also present the best-capacity configuration (M-20-BC) to evaluate the loss in capacity resulting from the other two configurations. Thus, we have three configurations for MEMS-based storage in total. Details of these configurations are in Table 6.1.

The best energy and performance configurations have a nominal throughput of 20 MB/s, whereas the SanDisk Standard CF card has a minimum read/write throughput of 10 MB/s[1]. This is an advantage for a MEMS-based storage device, since by deploying just a single chip a high throughput is attainable, whereas several Flash chips in addition to a high-end controller are needed to achieve such a throughput. Nevertheless, to enrich our comparison, we additionally select the overall best performance and energy configurations out of the configurations that employ just 2048 probes, which have a nominal throughput of 10 MB/s. The configurations are called M-10-BP and M-10-BE, respectively, and the settings are presented in Table 6.1. Next, we discuss the comparison in detail for the `scenarios` trace when formatting with the `ext3` file system and a 4 KB block size.

[1] Throughputs higher than 10 MB/s were observed for this sessionicular card type.



(a) Energy consumption



(b) Response time

Figure 6.1: Energy consumption and response time of the selected best-performance and best-energy configurations of our simulated MEMS-based storage device and the Flash card. The results correspond to simulations and measurement against the `scenarios` workload with the configuration `ext3-4KB`. The capacity of the devices is 2.60 GB (M-20-BE), 1.99 GB (M-20-BP), 2.60 GB (M-10-BE), and 2.27 GB (M-10-BP), respectively.

6.2.2 Results

Results for the `ext3-4K` Trace

Figure 6.1a and Figure 6.1b show the energy consumption and response time of the MEMS-based storage configurations, respectively. We exclude the best-capacity configuration (2.65 GB), since it exhibits a response time of approximately 116 ms, rendering it impractical.

Figure 6.1a shows that the Flash card consumes less energy than any configuration of the MEMS-based storage device. Figure 6.1b shows that in addition Flash outperforms all selected configurations. However, the difference in energy consumption between MEMS-based storage devices and Flash card is between 0% and 7%. The figure shows that the best-energy configurations consume the same amount of energy as the Flash card. The figure also shows the energy breakdown of the four MEMS-based storage devices and the Flash card. The prominent energy components of the MEMS-based storage device for any configuration are the read/write and inactive energy, like the Flash card.

The response time of MEMS-based storage devices varies greatly between configurations. The prominent component here is the read/write time, which varies from 2 – 6 ms; the seek time is in the range of 1.0 – 1.5 ms. Figure 6.1b shows that the M-20-BP configuration exhibits smaller read/write time than the Flash card. However, with the seek time added, the total response time becomes longer than that of the Flash card. The MEMS-based storage device has between 18% to 171% longer response time than the Flash card.

Results for the Other Traces

This section compares MEMS-based storage to Flash memory for the `scenarios` workload when formatting with different settings: `ext3-1KB`, `ext2-4KB`, and `ext2-1KB`. We select the overall-best performance configuration for each trace. The configurations are (4096,4,1), (4096,1,4), (4096,4,1), respectively. Figure 6.2a compares the energy consumption of these configurations with the Flash card. The figure shows that the MEMS-based storage device consumes between 4% to 22% more energy than the Flash card.

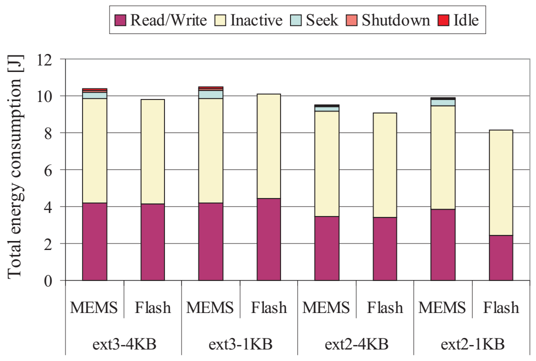
Figure 6.2b shows that MEMS-based storage devices exhibit shorter read/write time than the Flash card. However, with the seek time added, the response time of MEMS-based storage devices becomes longer. An exception exists for the `ext2-4KB` trace, whose corresponding best-performance MEMS-based storage device and the Flash card perform equally. Generally, MEMS-based storage devices exhibit between 0% and 31% longer response time.

6.3 Streaming Environments

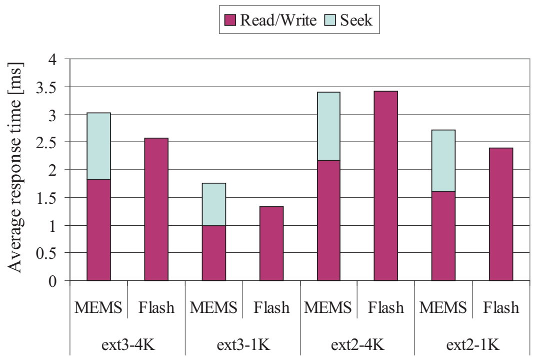
A streaming environment has a prevailing application type that runs all the time, which is the streaming application. We emulated a streaming environment on our PDA, and captured various streaming scenarios in the `multimedia` trace explained in Section 3.3.

6.3.1 Configurations of the Data Layout

From a storage perspective, streaming applications are characterized by three parameters: (1) streaming direction, (2) streaming bit rate, and (3) streaming (prefetching) unit. For example, a streaming application can stream from a storage device at a bit rate of 128 Kbps, reading 16 KB of data at a time.



(a) Energy consumption



(b) Response time

Figure 6.2: Comparison between the overall-best perforamnce configuration (M-20-BP) of our simulated MEMS-based storage device and the Flash card for the four traces of the `scenarios` workload

In streaming workloads, the streaming direction determines the operation (i.e., READ or WRITE), whereas the streaming bit rate with the streaming unit (henceforth the prefetching unit) determines the arrival time of requests. The streaming unit also determines the address and the size of the requests. The data layout of a MEMS-based storage device is influenced by the request address and size, and are thus influenced by the prefetching unit. Thus, the prefetching unit influences the data-layout of MEMS-based storage devices in streaming environments.

The influence of the first two parameters is rather straightforward, a storage device must sustain the streaming rate, and support streaming from and to it. This is achieved by increasing the number of parallel probes, and by allowing for read and write operations, respectively. On the other hand, the influence of the prefetching unit is subtle. We, therefore, limit our study in this work to studying the influence of the prefetching unit. We consider two

Table 6.2: Best-performance and best-energy configurations of the data layout of our simulated MEMS-based storage device

Session		# probes [probe]	Sector parallelism [sector]	Sector size [KB]
32 KB session	M-20-BP	4096	1	8
	M-20-BE	4096	32	4
128 KB session	M-20-BP	4096	2	4
	M-20-BE	4096	32	4
combined	M-20-BP	4096	1	4
	M-20-BE	4096	32	4

streaming sessions that stream from and to the storage device with a 32 KB and 128 KB prefetching unit, respectively. Both sessions are identical in terms of the number of streaming scenarios, their duration, and both stream at constant bit rates in the range of 32 – 512 Kbps.

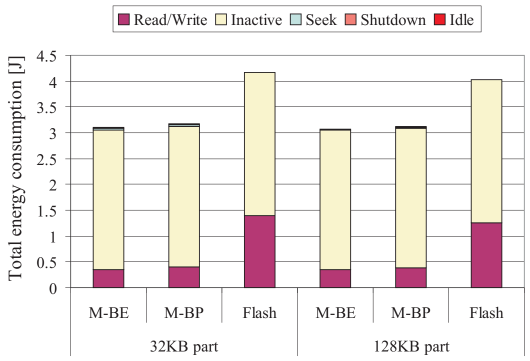
We carry out an exhaustive search to configure the three parameters of the data layout explained in Section 4.4.4. Table 6.2 lists the best-performance and best-energy configurations for each session of the `multimedia` trace that corresponds to the sessions, and for the sessions combined.

All configurations in Table 6.2 deploy 4096 probes, since increasing the number of active probes enhances the performance and energy-efficiency (see Section 4.4.3). The table shows that the best-performance configuration varies depending on the prefetching unit. We find multiple equivalent configurations in terms of energy consumption. We report in Table 6.2 the one that exhibits the best performance.

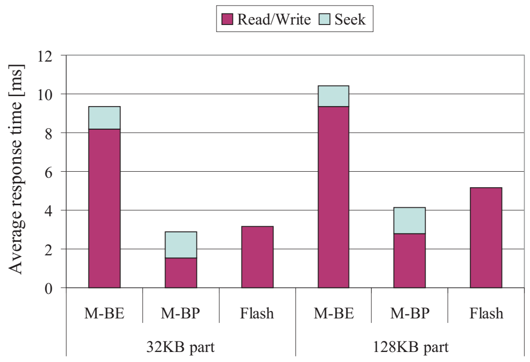
6.3.2 Results

Figure 6.3a shows the energy consumption and the energy breakdown of the two configurations of our MEMS-based storage device and the Flash card. In agreement with the results for the `scenarios` trace, the seek, shutdown and idle energy of MEMS-based storage devices are negligible. The main energy component of the MEMS-based storage device as well as the Flash card is again the inactive energy. This is because both technologies aggressively move into the inactive state, where they spend most of the time. The second energy component is the read/write energy.

Figure 6.3a shows that a MEMS-based storage device always consumes less energy than the Flash card for all scenarios. It shows that the best-energy configuration consumes between 24% to 26% less energy than the Flash card. Minimizing the energy consumption causes a loss in performance though.



(a) Energy consumption



(b) Response time

Figure 6.3: Total energy consumption and average response time of the best-energy and best-performance configuration of our simulated MEMS-based storage device and the Flash card for the 32 KB and 128 KB streaming sessions of the `multimedia` trace

Figure 6.3b shows that the Flash card has between 102% and 243% shorter response time than the best-energy configuration. The best-performance configuration of our MEMS-based storage device outperforms Flash on both accounts; it has at least 8% shorter response time and consumes at least 24% less energy. Reporting on the response time is not insightful for streaming applications, since throughput is the main concern. However, we do that due to the existence of filesystem requests (see Section 6.5.1).

To locate the cause of the poorer performance of the Flash card compared to the best-performance MEMS-based storage device, we study the distribution of the response time for every request size in the 32 KB and 128 KB sessions combined. Figure 6.4 summarizes the distributions by their average and standard deviation. It shows that the Flash card performs poorly for requests of size of 4 KB as the corresponding average indicates relative to other request

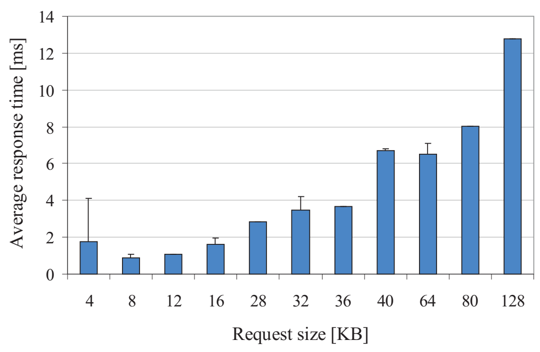


Figure 6.4: The average and the standard deviation of the response time of the Flash card for each request size of the 32 KB and 128 KB sessions combined. Some requests have zero standard deviation as for 128 KB request size.

sizes. The large standard deviation means a large variance in response time. Further, because of the asymmetric distribution of response time for the 4 KB requests, the standard deviation is larger than the mean. Note that other request sizes have poor performance too, such as the 40 KB request size.

Recall from Section 2.2.1 that internal data migration takes place in Flash to overwrite a page in a block. The migration process can take a relatively large amount of time, which explains the large response time for some request sizes. Other researcher [38] characterize the timing performance of Flash memory, and show a similar variation in performance.

6.4 Enhancing MEMS-Based Storage Devices

The comparison with Flash memory presents the need for enhancement of MEMS-based storage devices on the device level. We offer recommendations to the device designers to increase the viability of MEMS-based storage. The recommendations summarize lessons from our simulation results:

Probe data rate Increasing the attainable data rate per probe shortens the read/write time and reduces the energy consumption. Recall that read/write time and energy are the first and second prominent components, respectively. The read/write time shortens when using more probes per sector, whereas the read/write energy does not. Therefore, enhancing the inherent performance of the probe is necessary for energy efficiency.

Probe-field dimensions The seek time of a MEMS-based storage device constitutes a large session of the response time (Figure 6.2b). A MEMS-based storage device shuts down aggressively and, therefore, most of the seek times are in fact due to moving from the center (resting) position to the requested position. Reducing the dimensions of the probe field reduces the average traveled distance from the center, and thus reduces the seek time for the majority of requests.

Dynamic power net A dynamic power gating of probes, that allows to switch on and off (sets of) probes as needed, reduces the read/write energy (the second energy component). If unused probes can be switched off, we can efficiently implement large sector parallelism. This is sessionicularly important, because, as our results reveal, MEMS-based storage devices are configured with large sector parallelism to increase the effective capacity.

Actuators Section 4.4.3 shows that deploying a large number of probes reduces the actuation energy. Targeting Flash packages, that have lower power budget than CompactFlash like SD (Secure Digital) and MMC (Multimedia Card), limits the number of probes that can be deployed at a time. As a consequence, the actuation energy increases. For such small packages, energy-efficient actuators should be used.

Electronics MEMS-based storage devices can be shut down aggressively and thus spend a large fraction of the time in inactivity. Consequently, as Figure 6.2a shows, the inactive energy is the most prominent energy component. MEMS-based storage devices as well as Flash memories can reduce the inactive energy by using lower supply voltage, by applying voltage and frequency scaling techniques, or even by switching off most of the electronics.

6.5 Configuring for Streaming Applications

Streaming environments run a single specific application continually, namely the streaming application. Thus, the storage device predominately services requests, whose size is determined by the prefetching unit of the streaming application. This section investigates the influence of prefetching on the performance, energy consumption, and capacity of a MEMS-based storage device as secondary storage in mobile streaming systems. Specifically, we study:

- the influence of scaling up the size of the prefetching unit on the performance and energy consumption of a MEMS-based storage device (Section 6.5.2), and
- the direct configuration of the data layout of a MEMS-based storage device based on the prefetching unit of the streaming application (Section 6.5.3).

The two studies serve to determine the relation between the prefetching unit and the configuration of the data layout of a MEMS-based storage device. A good practical example, where understanding this relation can speed

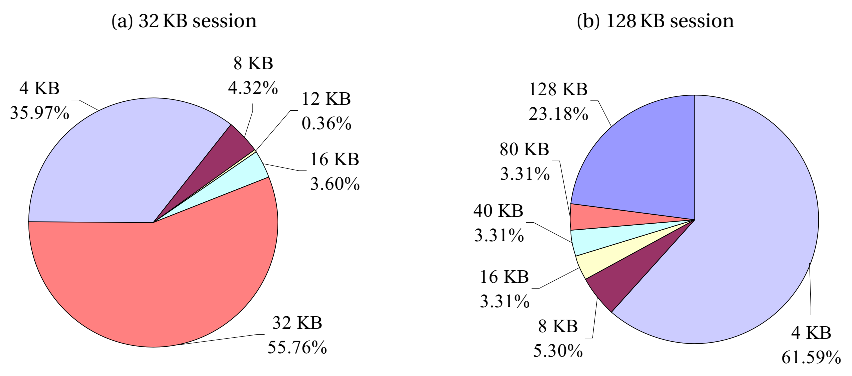


Figure 6.5: Breakdown of the request size for the 32 KB and 128 KB streaming sessions of the `multimedia` trace. Two main request sizes can be identified in each session, highlighting the bimodality of the distribution of the request size.

up the design process, is the design of streaming recording systems, such as the video camera. A MEMS-based storage device is suitable for video cameras, since it promises inexpensive green storage for such capacity-demanding applications. Before investigating the relation, we analyze the trace of the 32 KB session to examine the load a MEMS-based storage device services.

6.5.1 Bimodality of Request Size

Analyzing the 32 KB session (see Section 6.3.1), we make two observations. The first observation is that in addition to the stream of requests the streaming application exercises on the storage device, another stream exists due to the file system. Requests due to the file system are typically metadata writes and 4 KB in size. The second observation is that the I/O subsystem occasionally splits requests as shown in Appendix A.1. An application request is split into two I/O requests in order to maintain fairness between applications that compete for I/O. This sessionitioning results in smaller request sizes than is expected from the configured prefetching unit.

As a consequence, the storage device services two request streams of a different request size. Figure 6.5a presents the breakdown of the request size for the 32 KB session of the trace. The main stream of requests have a size of 32 KB, and makes up approximately 56% of the total number of requests. Requests of size of 4 KB make up approximately 36% of the total. These requests influence the configuration of the data layout, resulting in the product `sector parallelism` × `sector size` < 32 KB for the best-performance configuration for the 32 KB session (Table 6.2). We are interested in the `sector paral-`

`lelism × sector size` product, since it represents the minimum amount of data that can be accessed per request (of a streaming application for example). The number of simultaneously active probes depends mainly on the power budget of the package a device is housed in. For example, a SecureDigital (SD) package can afford a few hundred probes within its maximal 0.3 W power budget, whereas a CompactFlash (CF) card can afford a few thousand probes within a 1.1 W power budget.

The best-performance configuration is mainly influenced by the count of the requests, whereas the best-energy configuration is influenced by both the request size and the count of the requests. As a result, since large requests count for a large session of the total energy, their influence on the configuration is dominant. This explains why the best-energy configurations in Table 6.2 have a larger product (i.e., `sector parallelism × sector size`) than that of the best-performance ones.

Summarizing, a MEMS-based storage device experiences two streams of request size due to the streaming application and the file system, called bimodality of request size. Next, we discuss the influence of upscaling the request size of the streaming application.

6.5.2 Aggressive Prefetching

Streaming has a predictable nature, so that data can be prefetched to optimize for a certain resource. For example, large amounts of data are Prefetched from the disk drive, so that it spins off for a long period of time to save energy. In this section, we investigate the influence of scaling the prefetching unit on the performance and energy consumption of MEMS-based storage devices. We compare the configurations for the 32 KB session and the 128 KB session. Recall that both sessions are identical in the number of streaming scenarios and the streaming bit rates; they differ only in the size of the prefetching unit.

Increasing the prefetching unit results in fewer accesses to the storage device. Consequently, the seek energy (to seek from the center to the position of the requested data) and the shutdown energy (to bring the media sled back to the center) decrease. Since the device is accessed fewer times, it spends more time in inactivity. The first column in Table 6.3 indicates the effects of these direct influences.

For the best-performance configuration, increasing the prefetching unit results in a smaller product, namely `sector parallelism × sector size`. Table 6.2 shows that the sector size is 8 KB when the prefetching unit is 32 KB, whereas it is 4 KB when the prefetching unit is 128 KB. One would expect that the sector size increases when the prefetching unit increases, but this is not the case. The reason is that as the prefetching unit increases, requests due to the streaming application become fewer and larger, and thus the percentage of requests due to the file system increases. As a result, the influence of filesystem requests increases in determining the configuration that gives them the

Table 6.3: Effects of scaling up the prefetching unit of the streaming application on the configuration of MEMS-based storage devices. A "↑" denotes an increasing trend.

		prefetching unit		sector parallelism × sector size		overhead bits
trend		↑	⇒	↓	⇒	↑
time	read/write			↓		↑
	seek					
energy	read/write			↓		↑
	seek	↓				
	inactive	↑		↑		↓
	shutdown	↓				
	idle					
capacity						↓

best performance. Figure 6.5a and Figure 6.5b show that the percentage of requests of size of 4 KB increases from 36% for the 32 KB session to 62% for the 128 KB session, although their count are comparable (100 to 93 requests, respectively).

The second column in Table 6.3 shows the influence of decreasing the product on the response time and energy consumption of MEMS-based storage devices. Decreasing the product decreases the read/write time, and thus reduces the read/write energy. The reason is that when the sector size (or the sector parallelism) decreases, the utilization of probes increases, since probes more likely access useful data.

Reducing the sector size, however, results in more overhead bits per sector as shown in Section 4.4.2. As a result, the effective capacity of the device decreases as shown in third column in Table 6.3. Further, more time and energy are invested to read these bits. The overhead can be significant; the capacity of the best-performance configuration is 1.99 GB compared to 2.61 GB of the best-capacity configuration.

Figure 6.3a shows that the best-energy configuration of the MEMS-based storage device consumes comparable amounts of energy for the two prefetching units. This is because of the negligible seek and shutdown energy, so that savings on these components are unpronounced. Further, the energy consumed by the streaming requests is prominent and equal for either prefetching unit.

Summarizing, increasing the prefetching unit results in a smaller product: `sector parallelism × sector size`. A small product benefits small requests due to the file system but does not influence large requests due to the streaming application. Since a small product results in more overhead bits per sector, these overhead bits have a negative influence on the performance, energy-consumption and capacity. The overall influence on the performance and energy consumption is, however, proportional to the ratio of small requests to large requests. This ratio determines whether the overall effect is positive or negative.

6.5.3 Exploiting the Bimodality

We investigate the configuration of MEMS-based storage devices by exploiting the bimodality of the request size in streaming environments. We show that such a configuration methodology, based on the bimodality property, results in configurations that are close to the Pareto-optimal configurations found by an exhaustive search of the design space.

Recall from Section 4.4.3 that increasing the sector size directly reduces the overhead per sector, and enhances the performance and energy-efficiency. We investigate setting the sector size to one of the two main request sizes, while fixing the sector parallelism to one sector. We simulate for the 32 KB session and the 128 KB session. Figure 6.6 presents the improvement/degradation factor when configuring with one of the main request sizes for both sessions. The values are normalized to the best configuration for each respective design target for that session. That is, the energy figures for the two configurations of the 32 KB session, for instance, are normalized to the best-energy configuration for this session shown in Table 6.2. In contrast to energy consumption and response time, the lower the normalized value of the capacity, the smaller the capacity and thus the worse. The best overall configuration is the one that keeps the score on all targets close to one.

One observation is that the difference in energy consumption between the two configurations is negligible for both sessions. Figure 6.6 shows that the response time worsens drastically when formatting with a large sector size as for the 128 KB session; about 2.5 times longer response time than the best-performance configuration. The reason is that the increase in the sector size, increases the under-utilization of probes when servicing small requests. On the contrary, formatting with a large sector size reduces the overhead per sector, and thus increases the effective capacity.

Comparing the scores on the three targets of the 32 KB session and the 128 KB one reveals that aggressive prefetching on the application level leads to a loss in either the capacity as the case with (4096 probes, 1 sector, 4 KB) or the performance as the case with (4096 probes, 1 sector, 128 KB). Decreasing the prefetching unit enables the designers to reach smaller trade-offs between the capacity and the performance as the case with the configuration (4096 probes, 1 sector, 32 KB) for the 32 KB session (5% more energy, 11% longer response time, and 4% less capacity compared to the best configurations, respectively).

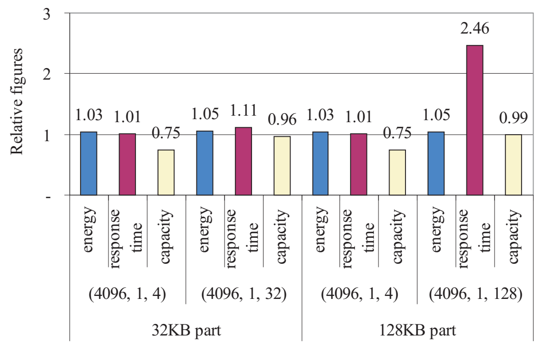


Figure 6.6: Possible configurations of a MEMS-based storage device based on the bimodal distribution of the request size in streaming environments. The sector size is set to one of the two main request sizes. The best overall configuration is the one that keeps the score on all targets close to 1, such as (4096, 1, 32).

In summary, designers of MEMS-based storage devices can exploit the bimodality property of streaming environments to find appropriate configurations *with small trade-offs quickly*. Further, decreasing the size of the prefetching unit of streaming applications enables the designer to find configurations that have smaller trade-offs between the design targets than what is possible with large prefetching units.

A Design Rule

The study of the configuration of the data layout boils down to the following design rule in streaming environments:

1. Model the workload as a composition of two streams with request sizes A and B.

2. Emulate the system with all configurations, whereby `number of active probes = maximum allowed`; and

   - `sector size` × `sector parallelism =` A
   - `sector size` × `sector parallelism =` B

   select the most beneficial configuration.

3. Scale down the prefetching unit to evaluate other possible trade-offs using the fast configuration method.

6.6 Summary

This chapter compares MEMS-based storage to Flash memory in two environments: (1) mixed-media and (2) streaming. Using the optimizations proposed in the previous two chapters we investigate how the performance, energy efficiency, and capacity of MEMS-based storage compare to Flash.

In the mixed-media environments, simulation results show that the best-energy configurations of a MEMS-based storage device consume about the same amount of energy as Flash memory. Other configurations consume up to 22% more energy than Flash memory. Simulation results reveal that a MEMS-based storage device performs as well as Flash in some cases, while it exhibits up to 31% longer response time than Flash.

In streaming environments, a MEMS-based storage device outperforms Flash. We observe that Flash memory exhibits bad performance for small write requests, because of the incurred internal data migration. Our modeled MEMS-based storage device has better performance for small write requests. Similar observations are made for the energy consumption.

The comparison study with Flash memory results in suggestions to enhance MEMS-based storage devices further. We identify some critical issues that require enhancement on the device level to elevate the timing performance and energy-efficiency of MEMS-based storage devices. These are the per-probe data rate, the dimensions of the probe field, the energy consumption of the actuators, and the powering net of the probes.

We present a quick and efficient methodology to configure MEMS-based storage devices for streaming applications. For configuration, the designer can exploit the bimodality of request sizes. The resulting configurations are very close to the Pareto-optimal configurations resulting from an exhaustive search, and exhibit smaller trade-offs between the design targets. Our experiment results suggest that configuring MEMS-based storage devices with a sector size larger than 4 KB and smaller than 128 KB leads to small trade-offs.

CHAPTER 7

Capacity-Demanding Applications

This chapter follows up on the study presented in the previous chapter, and investigates the utility of MEMS-based storage devices in mobile, capacity-demanding applications, such as portable video players and recorders. In this kind of applications, the Disk drive is more attractive than Flash memory from a cost perspective. However, energy efficiency is a concern, since these systems are battery powered.

We investigate three different streaming architectures that are optimized for energy efficiency. The first architecture is the conventional DISk-Based Architecture (**DISBA**). The energy saving of DISBA is, however, limited. We present two approaches to the energy-saving limitation of DISBA, one evolutionary, and another revolutionary approach. The evolutionary approach combines the Disk drive with Flash memory in a HYBrid-storage–Based Architecture (**HYBBA**). The revolutionary approach replaces the Disk drive with MEMS-based storage in a MEMs-storage–Based Architecture (**MEMBA**).

7.1 The DISBA Architecture

The conventional storage hierarchy, comprising a disk and DRAM (DISBA), accounts for a large proportion of the energy consumed by a computer system. For example, a spinning disk drive may account for as much as 30% of the total

Parts of this chapter have been presented at the 5-th IEEE/ACM/IFIP Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia'07), Salzburg, Austria [Khatib: 4]; and have been published in the Journal of Signal Processing Systems, Springer, 2008 [Khatib: 1].

energy consumed [80]. In streaming experiments in our laboratory, we found that a Microdrive plugged into an HP iPAQ H2215 PDA consumes as much as 23% of the total energy (see Figure 1.1a on page 2).

Many different solutions have been proposed to save on disk energy consumption. The basic idea, when operating the disk, is to spin off the disk drive for as long as possible and as often as possible. If the system workload is predictable, which is typically the case for predominantly streaming workloads, the disk can be spun off for long time periods, thus saving significantly on disk energy. In the conventional architecture (i.e., DISBA), Mesut *et al.* [81] prefetch streaming data into DRAM to maximize the spin-off period. However, DRAM dissipates a few milliwatts per megabyte to refresh (and thus retain) its contents. Hence, the maximum energy saving is attained when the sum of the disk energy and the DRAM retention energy is minimal. That is, the energy saving has an upper bound in DISBA.

Two factors influence the energy saving of DISBA. Firstly, if the streaming demand, represented by the number of concurrent streams and their bit rates, increases, the amount of DRAM needed for prefetching increases and so does its refresh energy. As a result, leaving the disk spinning all the time may even result in less total energy than prefetching. Secondly, if the form factor of the disk drive increases, the necessary amount of DRAM also increases to account for the larger latency due to the larger mechanical inertia. Both factors demand to increase the DRAM capacity and thus its retention energy, effectively lowering the energy saving bound to a point that renders prefetching impractical in DISBA.

7.2 The HYBBA and MEMBA Architectures

We present two architectures, one that can be realized with current technologies, and another that is based on future technologies. These are the HYBrid-storage–Based Architecture (HYBBA), and the MEMs-storage–Based Architecture (MEMBA), respectively. The basic idea of each architecture is as follows.

HYBBA We decouple DRAM from the disk drive to avoid the upper bound on the energy saving of DISBA. We interpose a NAND Flash memory between the Disk and the DRAM, where the Flash acts as a traffic reshaper. Flash memory does not require refresh cycles and the Flash size is limited only by its installation costs. Flash latency is lower than disk latency, but higher than DRAM latency, so a small DRAM buffer is sufficient to sustain the required streaming rates. The combination (Disk, Flash, and DRAM) makes optimal use of the Disk (by extending its spin-off period) and it makes optimal use of the DRAM (by using little of it). The rise in cost of the system through the addition of Flash can be balanced through a reduced DRAM capacity.

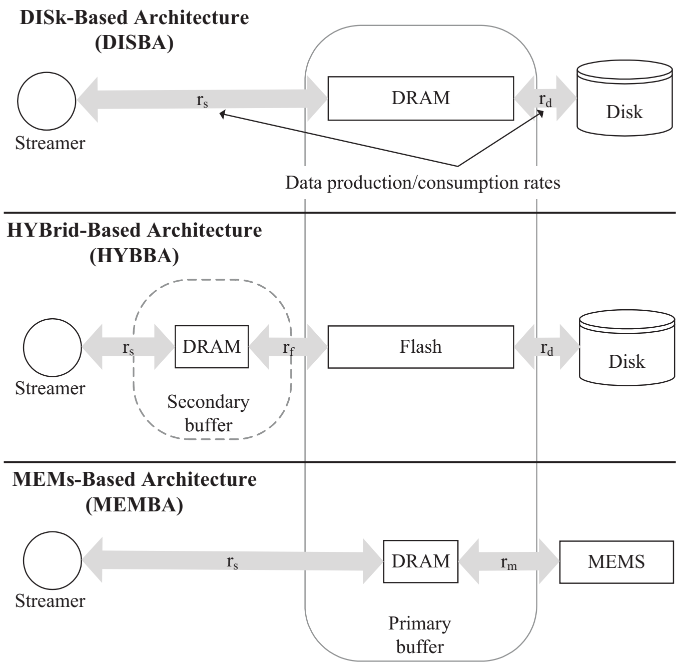

DISk-Based Architecture (DISBA)
Streamer
r_s
DRAM
r_d
Disk
Data production/consumption rates
HYBrid-Based Architecture (HYBBA)
Streamer
r_s
DRAM
r_f
Flash
r_d
Disk
Secondary buffer
MEMs-Based Architecture (MEMBA)
Streamer
r_s
DRAM
r_m
MEMS
Primary buffer

Figure 7.1: Block diagram of DISBA (*top*) that exists today, HYBBA (*middle*) that could be implemented today, and MEMBA (*bottom*) that can be implemented once MEMS-based storage devices become available.

MEMBA We replace the disk drive with MEMS-based storage to remove the upper bound on energy saving. MEMS-based storage promises high storage densities and is non-volatile, so that it can replace the disk drive. A MEMS-based storage device has no startup overhead (characteristic M1 in Section 2.1.4), and low shutdown overhead, so that frequent refills are possible. A MEMS-based storage device has short latency, so that a small DRAM buffer is sufficient to sustain the required streams. The capacity of MEMS-based storage is limited by the installation costs. The cost difference between MEMS-based storage and the Disk drive is a main concern for the feasibility of MEMBA. We contribute to reduce this cost as discussed in Sections 6.5.3 and 8.2. MEMS-based storage promises a smaller form factor, better shock resistance, and lighter weight than the Disk drive.

Figure 7.1 shows a block diagram of the three architectures. Both DISBA and MEMBA have a single buffering level, which is served by the DRAM. In contrast, HYBBA has two buffering levels. Flash serves as the primary buffer,

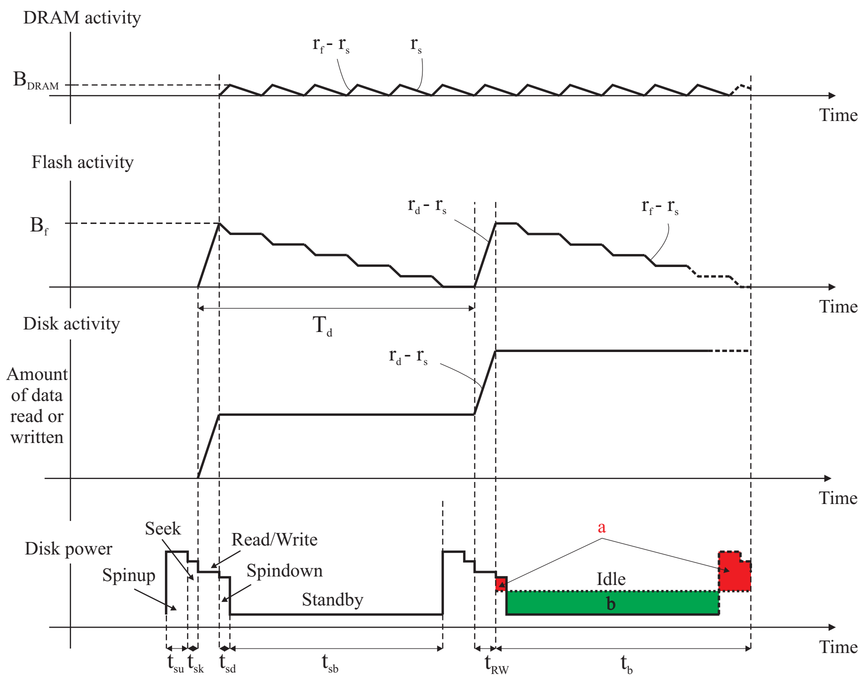


Figure 7.2: Activities of the disk drive, the Flash, and the DRAM in HYBBA during a refill cycle (T_d). The throughput of the Disk and the Flash are r_d, and r_f, respectively. The stream bit rate is r_s. Area "b" represent the amount of energy that can be saved if the disk is spun off, whereas area "a" represents the incurred overhead due to the subsequent spinup. If "b" > "a", energy can be saved by spinning off.

which communicates directly with the Disk. For performance reasons, a secondary level of buffering is needed to connect the Flash with the processor. DRAM serves as the secondary buffer in HYBBA.

7.3 Traffic Shaping

To save on its energy, a mechanical storage device (i.e., a Disk drive or a MEMS-based storage device) must shut down. To shut down a storage device, we must buffer data that should be written to or read from the device when it is in inactivity. Thus, a buffer is needed to host the required data temporarily. At the time the storage device is active, data can be moved between the buffer and the device. The buffering of data results in *shaping* the traffic from/to the storage device.

Figure 7.2 shows the shaping activity in the HYBBA architecture. The activity of the Disk, Flash, and DRAM are presented for two consecutive *cycles* (A cycle is the time that the system diverts from one specific state until it returns to that same state). To stream from the disk, the disk starts every cycle of T_d time units filling the Flash with a relatively large amount of data at a net rate $r_\mathrm{d} - r_\mathrm{s}$. In fact, the disk fills at r_d and the buffer empties at r_s at the same time.

At each cycle, the disk spins up and seeks before the actual refill. After the refill state the disk spins down immediately and remains in standby (i.e., spin-off state) to save energy. The Flash repeatedly refills the DRAM at rate $r_\mathrm{f} - r_\mathrm{s}$ with a relatively small amount of data, so that DRAM can be kept small to reduce its retention energy. When the Flash is nearly empty, the disk spins up, to prepare for the next cycle. To save energy, the refill cycle should be long enough to compensate for the energy consumed during spinup, seek, and spindown. Graphically, in Figure 7.2, area "b" should be larger than area "a".

Similar shaping activities take place in DISBA and MEMBA. The exception is that these architectures have only one buffering level. The activity of DISBA is similar to the Disk–Flash buffering activity, whereas MEMBA has a similar activity to the Flash–DRAM activity. MEMS-based storage is more similar to Flash in performance characteristics and buffering requirements than it is to the Disk.

The remainder of this chapter presents the results of evaluating and comparing the three architectures. We present a detailed study of the buffering requirement of each architecture and its energy saving. Then, we compare the energy saving of HYBBA and MEMBA with respect to the streaming demand and the best-effort demand. The final part studies the lifetime of the architectures. We present our methodology first.

7.4 Methodology

The remainder of this chapter presents the results of an analytical study. Appendix C gives the details of the underlying analytical models. The models compute the buffer capacities and the energy consumption for each of the three architectures.

Setup We consider a mobile streaming device that streams from a backing store. A Hitachi 1.8-inch Travelstar C4K40 HDD [82] serves as the backing store in DISBA and HYBBA. Our simulated MEMS-based storage device is the backing store of MEMBA. As a Flash memory in HYBBA, we use three SanDisk Extreme-III CompactFlash cards [83]. These cards operate in parallel to achieve a higher aggregate throughput than the disk, so that flash is not a bottleneck. In all architectures, Micron's DDR SDRAM [84] serves as a primary buffer in DISBA and MEMBA, and as a secondary buffer in HYBBA. Table 7.1 lists the settings of the disk drive, the flash memory, and the MEMS-based stor-

Table 7.1: Characteristics of the 1.8-inch disk, CF Extreme-III card, and the MEMS-based storage device

Parameter	1.8” HDD	Flash	MEMS	
throughput	187.2	240.0	160.0	[Mbps]
spinup power	1.485	–	–	[W]
seek power	1.122	–	0.672	[W]
access power	1.155	0.6	1.150	[W]
spindown power	0.330	–	0.672	[W]
idle power	0.330	–	0.120	[W]
standby power	0.099	0.005	0.005	[W]
spinup time	3.0	–	–	[s]
seek time	0.015	–	0.002	[s]
spindown time	0.5	–	0.001	[s]
overhead time	0.002	0.002	0.002	[s]

age device. The settings of the MEMS-based storage device are taken from our study in Section 4.2 and are presented by the PSM in Figure 4.3 on page 51.

Assumptions We assume that streams of video or audio are located on the middle tracks of the disk drive, thus we report on the average case here. In a separate study, we find that the difference in energy consumption between streaming from the outermost and the innermost tracks of a disk drive does not exceed 5% [Khatib: 1]. Further, we assume that the read throughput of the Flash is equal to the write throughput. We take into account the erasing overhead of Flash by assuming a low absolute throughput. In HYBBA, we fix the capacity of the secondary buffer of HYBBA to ten times the capacity of the real-time buffer, which is buffer that prevents under-run or overflow of streaming data to guarantee smooth streaming. As a result, the DRAM is not continuously refilling from Flash, which increases the availability of the I/O subsystem for other activities. As we will see shortly, even having a DRAM capacity that is one order of magnitude larger than the real-time buffer, the DRAM energy is four orders of magnitude smaller than the total energy of HYBBA, thanks to the presence of flash. We assume the maximum power dissipation of the MEMS-based storage device during idle or servicing modes. Thus, we report the worst-case energy consumption of MEMBA compared to HYBBA and DISBA.

7.5 Comparison

This section presents the results of a detailed analytical study of the buffering requirement and average power dissipation of each of the three architectures when streaming at a bit rate of 2048 Kbps, which corresponds to MPEG2 VCD quality.

7.5.1 Buffer Capacities

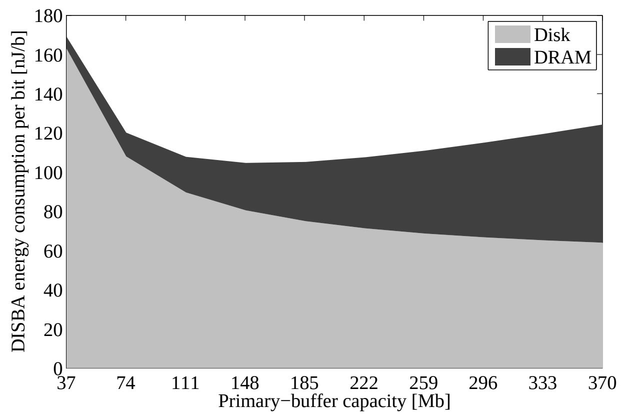
In DISBA, the minimum capacity of the DRAM is 37 Mb (mega**bit**), which is the capacity of the break-even buffer. The break-even buffer is the buffer that makes areas "a" and "b" equal in size (Figure 7.2), resulting in no energy saving. Increasing the primary buffer to capacities larger than the break-even capacity increases the energy saving. In HYBBA, the required Flash capacity is exactly the same as the DRAM in DISBA, since both are scaled based on the break-even buffer, which depends on the same disk drive used in either architecture. In HYBBA, the DRAM capacity reduces to approximately 4 Kb, a reduction by four orders of magnitude compared to the DRAM in DISBA. MEMBA uses one level of buffering, which is provided by DRAM. Due to the small overhead of MEMS-based storage, its primary buffer capacity is three orders of magnitude smaller than those of HYBBA and DISBA.

The DRAM capacities reported on here are (very) small compared to the units in which DRAM is available in the market. In practice, the small DRAM can be well substituted by an embedded memory, which is built in the processor chip. We, however, use DRAM in HYBBA and MEMBA to maintain a fair comparison to DISBA, which employs a large DRAM.

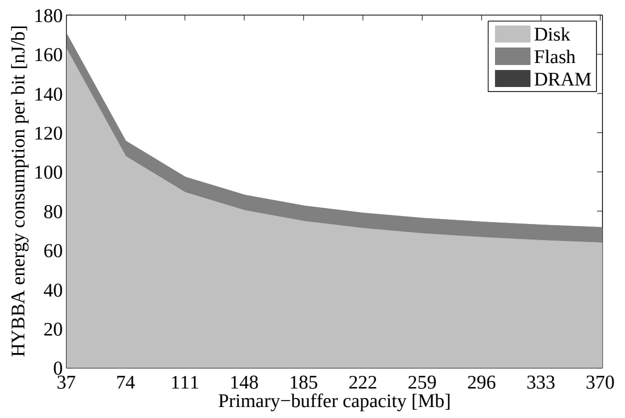
7.5.2 Energy Consumption

The primary buffer prefetches large amounts of data from the backing store, which is a disk drive in DISBA and HYBBA, and a MEMS-based storage device in MEMBA. Here, we compare the influence of upscaling the capacity of the primary buffer on the per-bit energy consumption. We scale the primary-buffer capacity up to ten times the break-even buffer capacity, and study the resulting per-bit energy consumption for each architecture. This range suffices to show the trends for each of the three architectures.

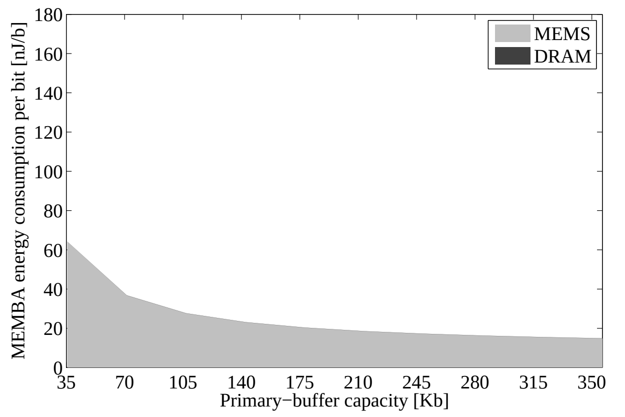
DISBA Figure 7.3a shows that by just using a DRAM capacity that is twice as large as the break-even buffer, the per-bit energy consumption drops about 50 nJ/b (approximately 35%). However, upscaling the buffer size does not always increase the energy saving. DISBA has a maximum energy saving at a DRAM capacity of 148 Mb= 4×37 Mb, where the total energy consumption is minimal. DRAM energy consumption worsens for large capacities, because DRAM retention energy increases proportionally to its increasing capacity.



(a) DISBA



(b) HYBBA



(c) MEMBA

Figure 7.3: Per-bit energy consumption of the three architectures, differentiated to components. Note the small capacity of the primary buffer for MEMBA (e.g., 35 Kb) relative to DISBA and HYBBA (e.g., 37 Mb).

HYBBA Figure 7.3b shows that HYBBA saving increases as the buffer capacity increases. However, as the capacity (of the primary buffer) increases, the number of spinups, seeks, and spindowns decreases, decreasing the amount of overhead energy that can be saved on. This corresponds to Amdahl's law, for which the power curve flattens for large capacities. The figure shows a constant contribution of flash to the total power distribution. This is not surprising, since increasing the mount of prefetching into flash capacity increases proportionally to the amount of time during which it sustains streaming, keeping the joule-per-bit ratio constant.

Increasing the capacity of the secondary buffer results in larger DRAM capacities, and thus more power dissipation. Although the DRAM capacity is scaled up ten times the real-time buffer in Figure 7.3b, its capacity remains in the order of tens of kilobits and its per-bit energy consumption is in the order of a tens of pico joules. In contrast, the total per-bit energy consumption is in the order of hundreds of nano joules. This explains why the DRAM energy contribution disappears in Figure 7.3b. HYBBA can drop below 70 nJ/b, whereas DISBA has a minimum power dissipation of approximately 100 nJ/b.

MEMBA Figure 7.3c shows that MEMBA saving increases as the buffer capacity increases. The larger the DRAM buffer, the larger the energy saving. However, as the capacity increases, the number of seeks, and shutdowns decreases, diminishing the return in energy saving. The MEMS-based storage device in MEMBA has low power profile compared to the disk drive. As a result, MEMBA consumes less than half the energy that HYBBA consumes. Further, the small overhead of MEMS-based storage allows to reduce the energy consumption of MEMBA up to five times less than HYBBA, while reducing the demand for buffering by three orders of magnitude; compare for example the energy consumption of HYBBA at 370 Mb to that of MEMBA at 350 Kb.

7.6 Energy-Efficiency of MEMBA

From the previous section, we find that HYBBA and MEMBA are promising alternatives to DISBA. This section studies the influence of the workload on the energy saving of MEMBA relative to HYBBA with respect to (1) the streaming demand, and (2) the best-effort demand. We investigate the energy-efficiency of MEMBA.

7.6.1 Streaming Demand

This section evaluates the energy saving of MEMBA relative to HYBBA for different streaming demands. The streaming demand is the number of concurrent streams that are played from/to the backing store, and the bit rate of each of these streams. Streams can be audio and video of various qualities, with

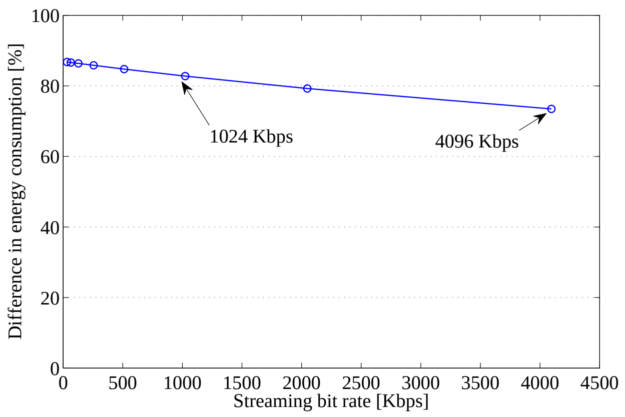


Figure 7.4: Relative difference in energy consumption between MEMBA and HYBBA ($\frac{E_{\text{HYBBA}}-E_{\text{MEMBA}}}{E_{\text{HYBBA}}}$) as a function of the streaming demand

corresponding different bit rates. We use the same hardware setup presented in Section 7.4.

We vary the bit rate in the range 32 – 4096 Kbps and select bit rates that are a power of two. The low bit rates correspond to audio, and the high bit rates correspond to video. We fix the capacity of the primary buffer to ten times the capacity of the break-even buffer in both architectures. Thus, the energy consumed by each architecture is mainly consumed during read/write and standby times. We calculate the relative difference ($\frac{E_{\text{HYBBA}}-E_{\text{MEMBA}}}{E_{\text{HYBBA}}}$) in energy consumption to quantify the extension in battery lifetime. E_{MEMBA} is the total energy consumed by the MEMS-based storage device and the DRAM in MEMBA during one refill cycle as calculated in Section C.4.

Figure 7.4 shows the difference in energy consumption between the two architectures. One observation is that MEMBA is at least 70% more energy efficient than HYBBA. The main reason is that its MEMS-based storage device dissipates significantly less power during standby compared to the disk drive in HYBBA. Remember that the energy due to startup and shutdown are negligible, since our primary buffer is ten times the break-even buffer (see Figure 7.3b and Figure 7.3c).

We observe that when the streaming demand increases, MEMBA becomes less efficient compared to HYBBA. This is explained by two facts. (1) MEMS-based storage is less energy-efficient than the disk drive in accessing sequential data. In Table 7.1 we see that the MEMS-based storage device dissipates a similar amount of access (or read/write) power as the disk drive but a MEMS-based storage device has a lower throughput. (2) Increasing the streaming de-

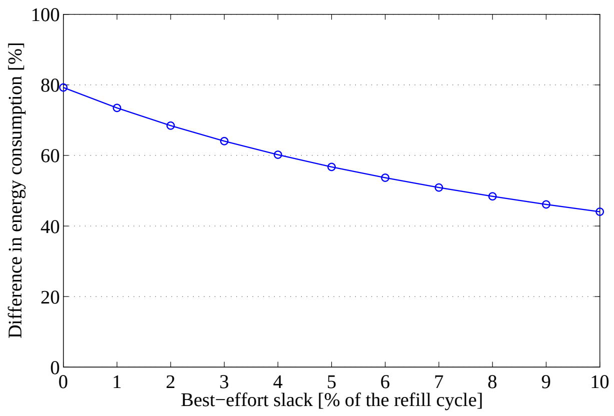


Figure 7.5: Relative difference in energy consumption between MEMBA and HYBBA ($\frac{E_{\text{HYBBA}}-E_{\text{MEMBA}}}{E_{\text{HYBBA}}}$) as a function of the best-effort slack

mand makes the access energy more prominent. From (1) and (2) we conclude that when the streaming demand increases the overall efficiency of MEMBA in accessing data decreases. Nevertheless, MEMBA is at least 70% more energy efficient than HYBBA.

7.6.2 Best-Effort Demand

In practice, streaming systems have to accommodate best-effort services for example to load applications and for paging. As a result, the backing store must support best-effort data in addition to the predominant streaming data. In Section C.5, we extend our analytical models to account for best-effort service. Best-effort service is provided during a scheduled **best-effort service period (or slack)** every refill cycle. We assume that the best-effort slack is smaller than or equal to 10% of the refill cycle.

We compare the energy saving of MEMBA relative to HYBBA for different best-effort demands. We vary the best-effort slack in the range 0% – 10% with an increment of 1%. We assume streaming at 2048 Kbps. Again, we compare when each architecture employs a primary buffer that is ten times the capacity of the break-even buffer. Thus, the energy consumption of either architecture is mainly consumed during read/write and standby times. We calculate the relative difference $\frac{E_{\text{HYBBA}}-E_{\text{MEMBA}}}{E_{\text{HYBBA}}}$.

Figure 7.5 shows the relative difference between MEMBA and HYBBA as a function of the best-effort slack. We observe that the energy efficiency of MEMBA decreases when the slack increases. Increasing the best-effort slack increases the amount of best-effort data accessed from the MEMS-based storage device. Further, the amount of streaming data prefetched in one refill cycle increases, because streaming should continue uninterruptedly during best-effort service. Recall that our MEMS-based storage device is less energy efficient than the disk drive at accessing data, so that MEMBA efficiency drops relative to HYBBA. Nonetheless, MEMBA is still at least 45% more energy efficient than HYBBA.

7.7 Reliability Study

Saving energy requires spinning the disk on and off repeatedly and overwriting the flash memory extensively in HYBBA. Likewise, the MEMS-based storage device in MEMBA seeks and shuts down repeatedly. This section studies the reliability of HYBBA in detail and follows up with a discussion of the reliability of MEMBA.

7.7.1 HYBBA

Compared to DISBA, HYBBA can reduce the number of cycles by exploiting larger buffers. Nonetheless, the disk drive and the flash memory lifetime remain important issues to guarantee a reliable system for the expected lifetime of the system. We give some background on disk and flash reliability first.

Background

Disk drive There are three methods to express disk drive reliability: (1) Mean Time To Failure (MTTF), (2) Mean Time Between Failures (MTBF), and (3) duty cycle rating; that is the number of times the drive can be spun down before the probability of failure on spin up becomes larger than 50%. The duty cycle ratio is more relevant than MTTF or MTBF in energy-conservative streaming architectures. Although an aggressively spun down disk drive saves energy, it results in an accelerated consumption of the duty cycles of the drive [85].

Disk manufacturers report on the duty cycle rating of their disk drives. The rating depends on the head parking mechanism implemented in the drive. There are two parking mechanisms, namely Contact Start/Stop (CSS) and the Ramp Load/Unload [64, Chapter 17, pages 636 – 637]. The Contact Start/Stop parks the head in the central zone of the platter, whereas the Ramp Load/Unload parks the head completely outside the platter area on a plastic ramp. The former mechanism is typically deployed in 3.5-inch disk drives, whereas the latter is typical in 1.8-inch and 2.5-inch drives, because of its better shock resistance. Duty cycles are in the range of 50,000 cycles for 3.5-inch disks and 500,000 cycles for 1.8- and 2.5-inch disks. The large difference is mainly due to stiction effects.

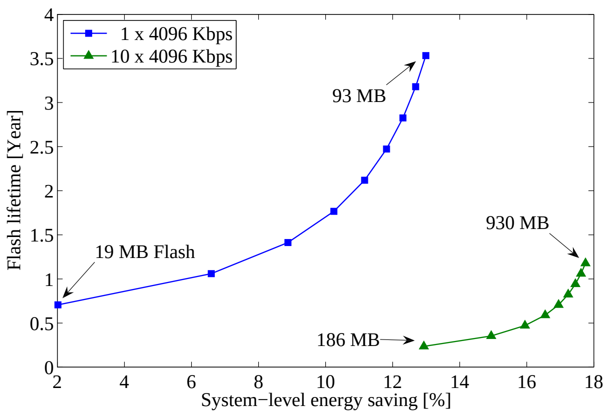


Figure 7.6: Flash lifetime versus system-level energy saving for a typical video-streaming workload and an extreme workload. The extreme workload of 10 × 4096 Kbps is presented to project the flash lifetime versus its capacity.

Flash memory Flash memory has limited endurance; a flash cell can be written for a fixed number of cycles ($10^5 - 10^6$ [86]). After that, its reliability to retain data drops drastically. To extend Flash lifetime, wear-leveling algorithms map writes to flash, so that all cells are rewritten the same number of times. When used as a FIFO (First In, First Out) buffer, wear leveling of flash is simply straightforward due to the inherent circular nature of the buffer refill. As such, the designer needs to mount enough flash capacity to meet a lifetime requirement, which is the concern with flash memory. We quantify the capacity in the following evaluation study.

Evaluation

We estimate the lifetime of the components of HYBBA by measuring their consumption of duty cycles. Every time the disk is spun up, one duty cycle is consumed from its lifetime and similarly from flash, since every disk spin up corresponds to a complete refill of the flash. We conduct a study to compare the energy-saving merit versus the flash lifetime. We use the same hardware setup as described in Section 7.4. We consider two workloads: (1) streaming one video at 4096 Kbps for 4 hours a day (e.g., serving a small household), and (2) an extreme workload of streaming ten videos at 4096 Kbps for 12 hours a day. Although the extreme workload is unrealistic for a portable device, we present it to estimate the maximum required flash capacity to meet a reasonable lifetime.

Figure 7.6 plots the flash lifetime versus the system-level energy saving of

HYBBA for the two workloads. Despite the extensive usage of flash in these workloads, a 100 MB and 1 GB flash suffices for 3.5 and 1.2 years, for the 1×4096 Kbps and 10×4096 Kbps workloads, respectively. A 1.8-inch disk drive will live three times as long as the flash, since its duty-cycle rating is 300,000 cycles (compared to 100,000 of flash). Note that the energy saving is taken to the maximum energy savings of DISBA for each workload. We present the saving in terms of the overall system saving to give the overall extension in the battery lifetime. To calculate the system-level saving, we assume that the storage hierarchy in DISBA consumes 33% respectively 50% of the total energy consumed by the system. This explains the larger saving for the second workload compared to the first one.

Figure 7.6 shows also that enlarging the flash capacity to achieve more energy saving increases its lifetime. In fact, we can see that after a certain flash capacity the energy saving starts saturating (i.e., little improvement), whereas a clear extension of flash lifetime (and thus system lifetime) is still possible. To match the expected lifetime of a mobile device, the designer needs to mount enough flash capacity into the system to guarantee an energy-efficient system for a desired lifetime. For example, assuming a mobile device is used for approximately 6 years, by just mounting a 200 MB flash, the system lives for 7 years with a 13% overall reduction in energy. A 5 GB flash is needed in the extreme case for a 6-year lifetime at an energy saving of 18%.

7.7.2 MEMBA

The lifetime of the MEMS-based storage device is a concern in MEMBA. Since the MEMS-based storage device is used as a backing store and not as a buffer, it is unlikely to face the endurance problem due to wear in the way flash memory wears in HYBBA. However, like the disk drive, the MEMS-based storage device in MEMBA is repeatedly shut down. In fact, the situation is even more crucial for the MEMS-based storage device than it is for the disk drive, because the buffer of the disk drive is three orders of magnitude larger than that of the MEMS-based storage device. As a consequence, to meet a required lifetime, a MEMS-based storage device should have a duty cycle rating that is three orders of magnitude larger than that of the disk drive, thus about 3×10^8 compared to 3×10^5 for the 1.8-inch disk drive.

Although a rating of 3×10^8 appears to be a large number, it is achievable for a MEMS-based storage device. Based on discussion with device designers, the following two reasons can be provided. Firstly, unlike the disk drive, a MEMS-based storage device has no rubbing surfaces that can wear each other, so that the motion components exhibit no tribology. Secondly, the springs are produced from a single crystal Silicon, which exhibits very small fatigue. Since the springs are operated within their designated operation range, fatigue is not a concern. Also, stopping the medium by shutting down is for the favor of increasing the lifetime of the springs. The previous two arguments are qualitative, and require a quantitative validation once a MEMS-based storage device is available. The duty cycle rating of 3×10^8 is a goal for the designers of MEMS-based storage devices to target for.

7.8 Summary

The conventional storage hierarchy (referred to as DISBA) composed of the Disk and DRAM saves on disk energy by buffering data in DRAM and spinning off the disk. The saving in DISBA is limited, since DRAM consumes energy to refresh its contents. This chapter presents two streaming architectures as an alternative for DISBA. The energy saving of the two architectures, which are referred to as HYBBA and MEMBA, increases as the amount of buffering increases, unlike DISBA.

HYBBA decouples the Disk from DRAM in DISBA by a Flash memory, so that the size of DRAM is reduced by four orders of magnitude. HYBBA presents a solutions that is realizable today by combining the existing Flash memory with the Disk drive in a Hybrid storage technology. In HYBBA, Flash is the main buffer of the disk drive, resulting in 15% energy saving compared to the optimal saving of DISBA.

MEMS-based storage promises high storage capacities, while exhibiting lower power profile compared to the disk. MEMBA employs future MEMS-based storage devices as a backing store to replace the Disk in DISBA. As a result, MEMBA is highly energy efficient. Our study shows that MEMBA can be as much as 80% more energy efficient than HYBBA, and it demands three orders of magnitude smaller DRAM capacity compared to a 1.8-inch disk drive.

Our study reveals that both HYBBA and MEMBA are promising solutions. Nonetheless, the feasibility of HYBBA and MEMBA depends mainly on the reliability of the flash and the MEMS-based storage device, respectively. For example, for the maximum energy saving in MEMBA, a MEMS-based storage device should have a duty cycle rating in the order of 10^8.

CHAPTER 8

Conclusions and Future Work

The research in this dissertation addresses an emerging storage technology, called MEMS-based storage. This technology is the basis of micro mechanical non-volatile storage devices, where a storage medium moves in the X and Y directions relative to a stationary array of thousands of read/write probes (or heads). MEMS-based storage is promising, since it offers storage devices that have a small form factor (41 mm^2), a high storage density (4 Tb/in^2), low energy consumption, and low per-bit cost.

To the best of our knowledge, this is the first work that tailors MEMS-based storage devices to mobile battery-powered devices. We propose and evaluate MEMS-based storage devices as secondary storage in two different mobile applications: capacity-modest applications, such as the handheld Personal Digital Assistant (PDA), and capacity-demanding applications, such as the portable video player. Currently, Flash and the Disk drive serve in these applications, respectively, where cost is the major discriminating factor.

In this work, we enhance MEMS-based storage devices to the level where serious deployment can be realized. This work addresses four types of policy: the power management policy, the shutdown policy, the data-layout policy, and the wear-leveling policy. We present at least one policy per type and assess its influence. We optimize for four design targets: energy consumption, capacity, response time, and lifetime of MEMS-based storage devices.

For capacity-modest applications, we prepared a complete PDA setup to record I/O traces for a mobile device to drive the simulation of a MEMS-based storage device. Prototypes of MEMS-based storage devices exist at present in research laboratories, but they are not available for public use. Therefore, we compare simulation results of a MEMS-based storage device to empirical data measured for a Flash card. The parameter settings of our simulated MEMS-based storage device are based on measurements from the IBM MEMS device. We studied mixed-media and streaming environments.

For capacity-demanding applications, we analytically evaluate the energy-saving merit and the lifetime of two new streaming architectures. The first architecture combines the disk drive, in the conventional architecture, with Flash. The second architecture replaces the disk drive with a MEMS-based storage device. We compare the two proposed architectures with respect to streaming bit rate and the amount of best-effort requests accommodated.

Next, we offer short answers to the three main questions of this research and in the next three sections we provide the answers in detail.

Q1: How can a MEMS-based storage device deliver appropriate quality of service: high energy-efficiency, high timing performance, large capacity, and long lifetime?

- Deploy power management to save 50% on the total energy consumption at an increase of 5% in response time.
- Exploit the springs to shut down efficiently and save 10% on the total energy consumption, while still reducing response time by a few percentages.
- Exploit knowledge of the application to increase the effective capacity by 15%.
- Deploying wear leveling increases the device lifetime by 46%.

Q2: How do MEMS-based storage devices compare to current storage devices composed out of Flash and the Disk drive?

- MEMS-based storage devices consume 20% more energy and exhibit 31% longer response time than Flash in mixed-media environments.
- If MEMS-based storage devices are optimized for energy `or` timing performance, they become comparable to Flash in mixed-media environments.
- MEMS-based storage devices consume 24% less energy and have 8% shorter response time than Flash in streaming environments.
- In predominately streaming environments, a MEMS–DRAM storage hierarchy consumes half the energy of a Disk–Flash–DRAM hierarchy. A MEMS–DRAM consumes one-fifth the energy of a Disk–DRAM hierarchy.

Q3: What are the components of a MEMS-based storage device that determine its quality of service?

- The recording technique (or the cost per bit) of MEMS-based storage must be significantly enhanced to become competitive with Flash and the Disk drive.

- The per-probe data rate must be enhanced to increase the throughput and reduce the read/write energy consumption.
- The number of probes per unit area must be increased to reduce the seek distance of a single probe and thus reduce the seek time.
- The springs and other mechanical components must bear a large number of on/off cycles in the order of 10^8 to allow for aggressive shutdowns.

8.1 System-Level Conclusions

In this work, we have devised policies to enhance the timing performance, energy-efficiency, capacity, and lifetime of MEMS-based storage devices. We address four types of policy, that if implemented, make MEMS-based storage competitive with Flash.

The first policy type is the power management policy, which decides on the time instance to shut down the device: returning the moving media sled to the resting position and parking it at this position. We studied the fixed-timeout power management policy for mixed-media environments. We showed that MEMS-based storage devices have no startup overhead and a small shutdown overhead, so that the fixed-timeout policy saves up to 95% of the idle energy, which makes up at least 40% of the total energy. The policy is near optimal, since a large saving is achieved at 4% degradation in performance. Further, our simulation results suggest avoiding immediate shutdowns, which reduces the response time up to 10%, since long seek distances are avoided.

The second type of policy is the shutdown policy, which decides on the method to shut down, which can either be passive (using the energy stored in the springs) or active (using the actuators). We devised an energy-efficient shutdown policy, which exploits the potential energy stored in the springs to drive the media sled toward its resting position. This policy allows to shutdown the device efficiently and aggressively. We devised a performance-efficient shutdown policy, which uses the actuators to drive the sled. Simulation results show that the energy-efficient policy can save up to 10% on energy compared to the performance-efficient. We found that the performance-efficient policy is more suitable for workloads with low locality and small inter-arrival times typically found in server workloads.

The third type of policy is the data-layout policy, which deals with striping sectors across the thousands of probes a MEMS-based storage device have. We formulated the data-layout with three parameters that need to be configured: the number of active probes, the number of sectors simultaneously accessible, and the size of the sector. We made the case for configuring these parameters by exploiting knowledge of the workload a MEMS-based storage device is expected to service. Such exploitation enhances the performance, and energy-efficiency, and also increases the effective capacity. Simulation results show enhancements in performance by at least 10%, reduction in energy by at least

5% at an increase of 15% in effective capacity relative to a simple configuration that employs all probes to access a single 512-byte sector.

The fourth type of policy addresses the lifetime of the individual probes in a MEMS-based storage device. Our study shows that some probes wear significantly (e.g., one order of magnitude) faster than others due to imbalance in distributing write requests across probes. Premature expiry of probes has severe consequences for a MEMS-based storage device, so that wear leveling is necessary. We devised an optimal wear-leveling policy that maximizes the lifetime by mapping sectors to probes in a round-robin fashion. This policy, however, increases the response time and the energy consumption by at least 20% relative to when deploying no policy. We devised two other policies that are deployed depending on the configuration of the data layout. Both policies approach the optimal policy very closely with respect to lifetime, whereas their timing performance and energy-efficiency are close to when no policy is employed. In a case study, we found that wear leveling increases the device lifetime by 46%.

8.2 Architecture-Level Conclusions

Following up the optimization studies, we study the integration of MEMS-based storage devices as secondary storage in the storage hierarchy of the computer system. We study two types of application: capacity-modest applications (e.g., a PDA) and capacity-demanding applications (e.g., a portable video player).

For capacity-modest applications, our study shows that a MEMS-based storage device consumes up to 22% more energy than Flash and exhibits a 31% longer response time. However, we show that configurations of MEMS-based storage devices tailored toward minimizing response time or energy consumption perform comparably to or consume as much energy as Flash. We also study streaming environments in capacity-modest applications. Comparison shows that internal migration of data in Flash places Flash behind MEMS-based storage on the energy-efficiency scale by at least 8%. Results also show that a proper configuration exhibits a large sector size (e.g., 16 KB), so that the capacity increases by a factor 1.28 compared to sector size of 512 B. Figure 8.1 visualizes our contribution to enhance the energy-efficiency and reduce cost of MEMS-based storage devices.

For capacity-demanding applications, we find that MEMS-based storage devices promise a good energy-efficient alternative to disk drives. In comparison with a Disk–Flash–DRAM hierarchy, we find that a MEMS–DRAM hierarchy is at least 70% more energy efficient. Allowing best-effort service for up to 10% of the time, we find that the MEMS–DRAM hierarchy is still at least 40% more energy efficient. However, for such enhancement in energy efficiency, a MEMS-based storage device must be able to switch on and off three

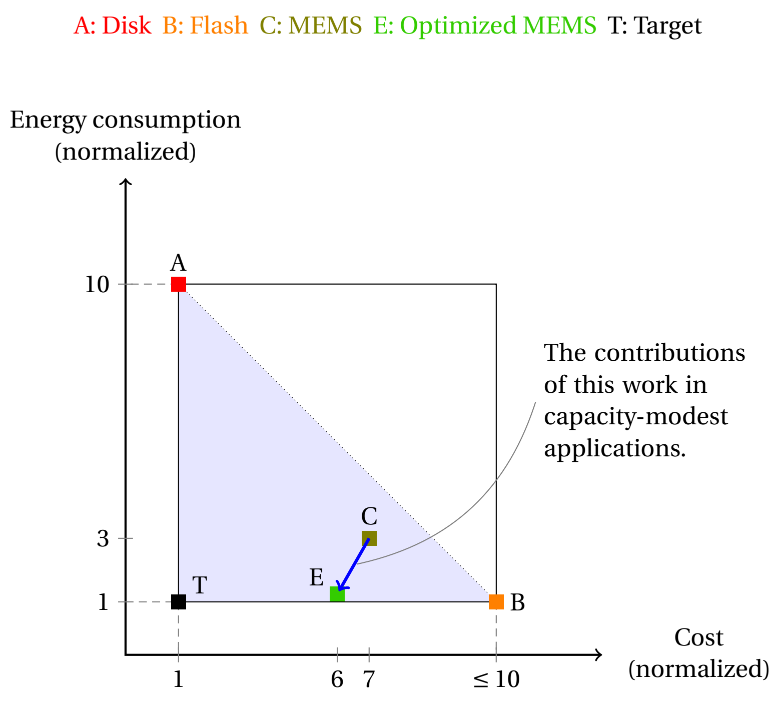


Figure 8.1: A visualization of the relative position of each of the addressed storage technologies for capacity-modest applications. A typical MEMS-based storage device (C) consumes approximately three times more energy than a Flash (B), and is presumably seven times more expensive than the Disk drive. Enforcing the energy-enhancement policies, an optimized MEMS-based storage device (E) is as energy efficient as a Flash. The data-layout policy increases the effective storage capacity, and thus reduces the per-bit cost by a factor of 1.15 from seven times down to six times the cost of the disk drives. This figure presents a conservative view and shows E for the minimum improvement factors in order to arrive at safe conclusions. That is, larger improvements are absolutely possible. Note that point A in handheld mobile devices is represented by the Microdrive, which is obsolete at the moment. We, however, present it for the sake of completeness. The figure is for devices with a CompactFlash (CF) format.

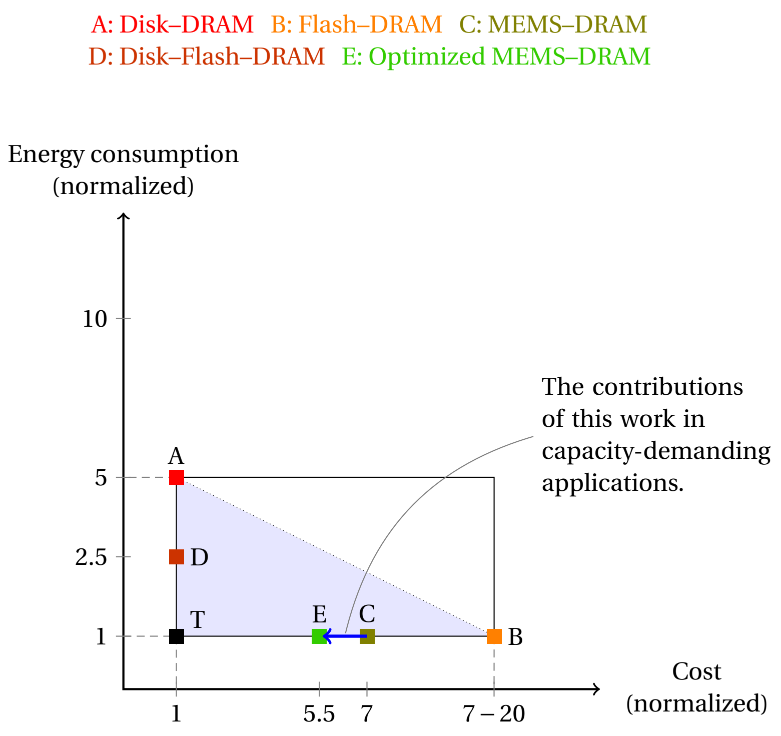


Figure 8.2: A visualization of the relative position of each of the addressed storage technologies for capacity-demanding applications. The points position the storage technologies when applying cross-layer optimization in a predominately streaming environment. If a MEMS-based storage device implements power management, a MEMS–DRAM hierarchy (C and E) becomes as energy efficient as a Flash–DRAM hierarchy (B). Our results show that a Disk–Flash–DRAM hierarchy (D) consumes at least 2.5 times more energy than a MEMS–DRAM hierarchy. An optimized Disk–DRAM hierarchy consumes twice as much as a Disk–Flash–DRAM hierarchy. Because of the reduction in DRAM capacity, a Disk–Flash–DRAM hierarchy costs as much as a Disk–DRAM hierarchy (including doubling the Flash capacity for a 6-year lifetime requirement), whereas a MEMS–DRAM hierarchy (C) costs seven times as much. The reduction in DRAM capacity has marginal influence on the reduction of the large difference in cost (i.e., seven times) between a MEMS-based storage device and a disk drive. This is because saving a few orders of magnitude on DRAM capacity has a marginal influence on reducing the cost of the large capacities demanded in the backing store, which is in the order of 10^{12} bits. Borrowing the results of our work for streaming in capacity-modest applications, we show an increase in the effective capacity by 28%, reducing the cost from 7 times (C) to 5.5 times (E) the disk drive. The figure is for devices with a 1.8-inch format.

orders of magnitude more often than a 1.8-inch disk drive; that is about 10^8 times. The realization of the MEMS–DRAM hierarchy depends on the MEMS–to–Disk cost ratio. Figure 8.2 visualizes our contributions to enhance the energy efficiency of the computer system with several hierarchies. It also shows the reduction in the per-bit cost for the MEMS–DRAM hierarchy.

8.3 Device-Level Recommendations

Our optimization, integration, and comparison studies reveal several potential enhancements for MEMS-based storage devices. Further, we recommend the implementation of mechanisms to enforce the policies devised in this dissertation. We categorize our recommendations based on the design targets:

ENERGY CONSUMPTION AND THROUGHPUT:

Probe data rate Increasing the attainable data rate per probe shortens the read/write time and reduces the energy consumption. The read/write time and energy are the first and second prominent components, respectively. The read/write time decreases when using more probes per sector, whereas the read/write energy is not influenced. Therefore, enhancing the inherent performance of the probe is necessary for energy efficiency. Further, higher data rates are attainable.

RESPONSE TIME:

Probe-field dimensions The seek time of a MEMS-based storage device constitutes a large part of the response time. A MEMS-based storage device shuts down aggressively and, therefore, most of the seek times are in fact due to moving from the center (resting) position to the requested position. Reducing the dimensions of the probe field reduces the average traveled distance from the center, and thus reduces the seek time for the majority of requests.

ENERGY CONSUMPTION:

Dynamic power gating Employing a dynamic gating to power probes allows to switch (sets of) unused probes on and off on demand. This reduces the read/write energy (the second most important energy component). If unused probes can be switched off, we can efficiently implement large sector parallelism. This is particularly important, because, as our results reveal, MEMS-based storage devices are best configured with large sector parallelism to increase the effective capacity.

Probe power dissipation A large number of probes are deployed to elevate the throughput of a MEMS-based storage device. This results in a relatively large power budget that is required to power on the probes, approximately 1 W in our simulated device. The probe power dissipation should be reduced in order to target mobile devices as well as to provide SSD-like devices. The reduction in probe power directly reduces the read/write energy, which constitutes a prominent component of the total energy.

Actuators Deploying a large number of probes reduces the actuation energy. Targeting Flash packages, that have smaller power budget than CompactFlash (CF), such as SD (Secure Digital) and MMC (Multimedia Card), limits the power budget and thus the number of probes that can be deployed simultaneously. As a consequence, the actuation energy increases. For such small packages, energy-efficient actuators are needed.

Electronics To save energy, MEMS-based storage devices shut down aggressively, and thus spend a large fraction of the time in inactivity. Consequently, the inactive energy is a significant energy component. MEMS-based storage devices as well as Flash memories can reduce the inactive energy by using lower supply voltage, by applying voltage and frequency scaling techniques, or even by switching off most of the electronics.

Power State Machine (PSM) Power management is an efficient way to reduce the energy consumption of a MEMS-based storage device. To enforce the power management policy, a MEMS-based storage device must employ a Power State Machine (PSM) to track its operation modes. As a result, the device can decide on the time instance to shut down in order to save energy.

COST AND CAPACITY:

Recording technique We proposed a method to format the data layout of MEMS-based storage devices to increase the effective capacity and thus reduce the per-bit cost. Factors up to 1.3 reduction in cost have been demonstrated. The system-level reduction of cost remains, however, limited, and, therefore, increasing the storage density on the device level is absolutely needed. Several techniques can be looked into to increase the density. It is important to reduce the cost by a factor of five at least based on our assumption that a MEMS-based storage device costs seven times as much as a disk drive.

Duty cycle rating To save energy, a MEMS-based storage device employs a small streaming buffer. Consequently, it has to fill the buffer frequently. The frequent number of refills results in a large number of transitions between active (moving) and inactive (stationary) modes. The number of transitions is in the order of 10^8 cycles for a mobile device of a typical lifetime of approximately 6 years. Critical components of a MEMS-based storage device, such as the springs and/or their fixing points, must be immune to fatigue to realize the target of 10^8 cycles.

MEMS Translation Layer (MTL) Wear leveling is necessary for MEMS-based storage devices to guarantee an economical and healthy life of the device. Wear leveling involves: dynamic mapping, tracking of wear across probes, and keeping track of the mappings. A MEMS Translation Layer (MTL) is needed to carry out these tasks. MTL can well be implemented in the firmware of a MEMS-based storage device.

8.4 Future Work

Several research directions can be identified that follow logically from the work addressed in this dissertation. In the following, we discuss some of these directions.

Archiving applications Archiving systems require high-capacity and energy-efficient storage devices. Clustering several MEMS-based storage modules into SSD-like packages can provide the required storage. One aspect to look into is the deployment of log-structured file systems. An arising research question is: *What does probe wear-leveling and the data layout look like under log-structured file system?*

Multi-sled devices This work focuses on enhancing single-sled MEMS-based storage devices for mobile applications. Multi-sled MEMS-based storage devices, however, promise more freedom in servicing I/O requests and thus can be more energy-efficient. An interesting research question is: *Which trend should MEMS-based storage follow: increasing the number of probes per sled, increasing the number of sleds per the same number of probes, or perhaps a bit of both; and under what conditions?*

Static actuators Our simulated MEMS-based storage device has electromagnetic actuators, which dissipate a reasonable amount of power. Another type of actuator is needed particularly for power-limited packages, such as SecureDigital (SD). Unlike electromagnetic actuators, electrostatic actuators dissipate no power to keep the sled stationary and dissipate a small amount of power during motion. A research question is: *What are the effects on the power management, shutting down, and the data layout when employing electrostatic actuators?*

Medium wear-leveling Medium wear-leveling becomes an issue only when probe endurance is at least two orders of magnitude larger than medium endurance. The reason is that consuming one cycle of sector endurance corresponds to consuming tens (or hundreds) of cycles of probe endurance, since a probe writes tens of bits per sector due to striping. If the medium wear arises as a challenge, a research question is: *How can we effectively address probe and medium wear combined, so that the quality of service of a MEMS-based storage device is minimally affected?*

List of Acronyms

ADC	Analog-to-Digital Converter
AFM	Atomic Force Microscopy
CD-ROM	Compact-Disk Read-Only Memory
CF	CompactFlash
DAQ	Data Acquisition
DRAM	Dynamic Random Access Memory
ECC	Error-Correction Code
EEPROM	Electrically Erasable Programable Read-Only Memory
FCFS	First-Come, First-Served
LBA	Logical Block Address
PBA	Physical Block Address
ROM	Read-Only Memory
SSD	Solid-State Disk
SDF	Shortest Distance First
HDD	Hard Disk Drive
MEMS	Micro-Electro-Mechanical Systems
µSPAM	Micro Scanning Probe Array Memory
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MTL	MEMS Translation Layer
NVRAM	Non-Volatile Random Access Memory
PDA	Personal Digital Assistant
PIO	Programmed Input/Output
PM	Power Management
PSM	Power State Machine
RF-MEMS	Radio-Frequency Micro-Electro-Mechanical Systems
RAID	Redundant Array of Inexpensive (or Independent) Disks

OPIE	Open Palm Integrated Environment
SPM	Scanning Probe Microscopy
SPTF	Shortest Positioning Time First
STM	Scanning Tunneling Microscopy
ZSPTF	Zone-Based Shortest Positioning Time First

REFEREED PUBLICATIONS BY THE AUTHOR

[Khatib–1] M. G. Khatib, P. H. Hartel, and H. W. van Dijk, "Energy-Efficient Streaming Using Non-volatile Memory," *Journal of Signal Processing Systems*, 2008. Published online at `http://dx.doi.org/10.1007/s11265-008-0308-1`.

[Khatib–2] M. G. Khatib and P. H. Hartel, "Power Management of MEMS-Based Storage Devices for Mobile Systems," in *Proceedings of the 8-th ACM & IEEE International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'08), Atlanta, Georgia, USA*, pp. 245–254, October 2008.

[Khatib–3] M. G. Khatib, E. L. Miller, and P. H. Hartel, "Workload-Based Configuration of MEMS-Based Storage Devices for Mobile Systems," in *Proceedings of the 8-th ACM & IEEE International Conference on Embedded Software (EMSOFT'08), Atlanta, Georgia, USA*, pp. 41–50, October 2008.

[Khatib–4] M. G. Khatib, B. J. van der Zwaag, P. H. Hartel, and G. J. M. Smit, "Interposing Flash Between Disk and DRAM to Save Energy for Streaming Workloads," in *Proceedings of the 5-th IEEE/ACM/IFIP Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia'07), Salzburg, Austria*, pp. 7–12, October 2007.

[Khatib–5] M. G. Khatib, B. J. van der Zwaag, F. C. van Viegen, and G. J. M. Smit, "Striping Policy as a Design Parameter for MEMS-Based Storage Systems," in *Proceedings of the 2-nd International Workshop on Software Support for Portable Storage, Seoul, Korea*, October 2006.

[Khatib–6] P. H. Hartel, L. Abelmann, and M. G. Khatib, "Towards Tamper-Evident Storage on Patterned Media," in *Proceeding of the 6-th USENIX Conference on File and Storage Technologies (FAST'08), San Jose, California, USA*, pp. 283–296, February 2008.

Non-Refereed Technical Reports

[Khatib–7] J. B. C. Engelen and M. G. Khatib, “A ’Millipede’ scanner model - Energy consumption and performance,” Technical Report TR-CTIT-08-51, University of Twente, Enschede, July 2008.

[Khatib–8] M. G. Khatib, J. B. C. Engelen, and P. H. Hartel, “Shutdown Policies for MEMS-Based Storage Devices – Analytical Models,” Technical Report TR-CTIT-08-03, University of Twente, Enschede, January 2008.

LIST OF PUBLICATIONS

[1] R. S. Williams, “How We Found the Missing Memristor.” http://www.spectrum.ieee.org/dec08/7024. Accessed in February 2009.

[2] J. Patzakis, “New accounting reform laws push for Technology-Based document retention practices,” *Int. Journal of Digital Evidence*, vol. 2, p. paper 2, Spring 2003.

[3] M. Taylor, “The EU data retention directive,” *Computer Law & Security Report*, vol. 22, no. 4, pp. 309–312, 2006.

[4] “Five strategies for cutting data center energy costs through enhanced cooling efficiency.” http://www.liebert.com/common/ViewDocument.aspx?id=13. Accessed in April 2008.

[5] G. Basheda, M. W. Chupka, P. Fox-Penner, J. P. Pfeifenberger, and A. Schumacher, “Why are electricity prices increasing?,” tech. rep., The Brattle Group, June 2006.

[6] R. F. Freitas and W. W. Wilcke, “Storage-class memory: The next storage system technology,” *IBM Journal of Research and Development*, vol. 52, pp. 439–447, July/September 2008.

[7] S. K. Lai, “Flash memories: Successes and challenges,” *IBM Journal of Research and Development*, vol. 52, pp. 529–535, July/September 2008.

[8] K. Prall, “Scaling non-volatile memory below 30nm,” *Non-Volatile Semiconductor Memory Workshop, 2007 22nd IEEE*, pp. 5–10, August 2007.

[9] R. Bennewitz, J. N. Crain, A. Kirakosian, J. L. Lin, J. L. McChesney, D. Y. Petrovykh, and F. J. Himpsel, “Atomic scale memory at a silicon surface,” *Nanotechnology*, vol. 13, pp. 499–502, August 2002.

[10] M. A. Lantz, H. E. Rothuizen, U. Drechsler, W. Häberle, and M. Despont, “A Vibration Resistant Nonopositioner for Mobile Parallel-Probe Storage Applications,” *Journal of Microelectromechanical Systems*, vol. 16, pp. 130–139, February 2007.

[11] "Nanochip develops MEMS-based storage.." `http://nanochipinc.com/tech.htm`. Accessed in November 2007.

[12] G. Binnig, C. F. Quate, and C. Gerber, "Atomic Force Microscope," *Physical Review Letters*, vol. 56, pp. 930–933, March 1986.

[13] G. Binnig, H. Rohrer, C. Gerber, and E. Weibel, "Tunneling through a controllable vacuum gap," *Applied Physics Letters*, vol. 40, pp. 178–180, January 1982.

[14] L. R. Carley, J. A. Bain, G. K. Fedder, D. W. Greve, D. F. Guillou, M. S. C. Lu, T. Mukherjee, S. Santhanam, L. Abelmann, and S. Min, "Single-chip computers with microelectromechanical systems-based magnetic memory," *Journal of Applied Physics*, vol. 87, pp. 6680–6685, May 2000.

[15] D. Briere and P. Hurley, "MEMS makes headway in consumer market." `http://www.eetimes.com/showArticle.jhtml;jsessionid=KIPMMHYW2KQ4KQSNDLQCKHSCJUNN2JVN?articleID=210605037`. Accessed in February 2009.

[16] S. D. Senturia, *Microsystem Design.* Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[17] R. C. Johnson, "RF-MEMS aims to tune mobile wireless." `http://www.eetimes.com/showArticle.jhtml?articleID=203100029`. Accessed in February 2009.

[18] M. Patrascu, *Characterization, modeling and control of the* μ*Walker - a micro actuator for data storage.* PhD thesis, University of Twente, Enschede, September 2006.

[19] T. Thomson, L. Abelmann, and J. P. J. Groenland, "Magnetic data storage: past, present and future," in *Magnetic Nanostructures in Modern Technology*, NATO Science for Peace and Security Series Subseries: NATO Science for Peace and Security Series B: Physics and Biophysics, pp. 237–306, Berlin: Springer Verlag, 2007.

[20] H. J. Mamin and D. Rugar, "Thermomechanical writing with an atomic force microscope tip," *Applied Physics Letters*, vol. 61, pp. 1003–1005, August 1992.

[21] Y. Cho, S. Hashimoto, N. Odagawa, K. Tanaka, and Y. Hiranaga, "Realization of 10 $\mathrm{Tbit/in}^2$ memory density and subnanosecond domain switching time in ferroelectric data storage," *Applied Physics Letters*, vol. 87, December 2005.

[22] R. C. Barrett and C. F. Quate, “Charge storage in a nitride-oxide-silicon medium by scanning capacitance microscopy,” *Journal of Applied Physics*, vol. 70, pp. 2725–2733, September 1991.

[23] H. F. Hamann, M. O’Boyle, Y. C. Martin, M. Rooks, and H. K. Wickramasinghe, “Ultra-high-density phase-change storage and memory,” *Nature Materials*, vol. 5, pp. 383–387, May 2006.

[24] M. Bolks, F. Hanssen, L. Abelmann, P. Havinga, P. Hartel, P. Jansen, C. Lodder, and G. Smit, “Micro scanning probe array memory (μspam),” in *2nd PROGRESS Workshop on Embedded Systems*, (Veldhoven, the Netherlands), October 2001.

[25] E. Sarajlic, *Electrostatic microactuators fabricated by vertical trench isolation technology*. PhD thesis, University of Twente, Enschede, May 2005.

[26] A. G. van den Bos, *CantiClever: Planar fabrication of probes for magnetic imaging*. PhD thesis, University of Twente, Enschede, November 2003.

[27] A. J. le Fèbre, *Field emission sensing for non-contact probe recording*. PhD thesis, Univ. of Twente, Enschede, March 2008.

[28] R. M. Vallejo, *Magnetic media patterned by laser interference lithography*. PhD thesis, University of Twente, Enschede, April 2006.

[29] A. Pantazi, A. Sebastian, T. A. Antonakopoulos, P. Bächtold, A. R. Bonaccio, J. Bonan, G. Cherubini, M. Despont, R. A. DiPietro, U. Drechsler, U. Dürig, B. Gotsmann, W. Häberle, C. Hagleitner, J. L. Hedrick, D. Jubin, A. Knoll, M. A. Lantz, J. Pentarakis, H. Pozidis, R. C. Pratt, H. Rothuizen, R. Stutz, M. Varsamou, D. Wiesmann, and E. Eleftheriou, “Probe-based ultrahigh-density storage technology,” *IBM Journal of Research and Development*, vol. 52, pp. 493–511, July/September 2008.

[30] D. Wiesmann, U. Dürig, B. Gotsmann, A. Knoll, H. Pozidis, F. Porro, and R. Vecchione, “Ultra-high storage densities with thermo-mechanical probes and polymer media,” in *the Innovative Mass Storage Technologies Workshop (IMST’07)*, (Enschede, Netherlands), pp. 19–20, June 2007.

[31] Y.-S. Kim, S. Jang, C. Lee, W.-H. Jin, I.-J. Cho, M.-H. Ha, H.-J. Nam, J.-U. Bu, S.-I. Chang, and E. Yoon, “Thermo-piezoelectric Si_3N_4 cantilever array on cmos circuit for high density probe-based data storage,” *Sensors And Actuators A*, vol. 135, pp. 67 – 72, March 2007.

[32] H. Shin, J. U. Jeon, S. Hong, J. joon Choi, H.-M. Jeong, D.-K. Min, Y. I. Kim, C. S. Lee, and S. Lee, “Construction of probe-based data storage with ultra-high areal density,” *IEEE-NANO 2001. Proceedings of the 2001 1st IEEE Conference on Nanotechnology, 2001.*, pp. 207–212, October 2001.

[33] J. B. C. Engelen, H. E. Rothuizen, U. Drechsler, R. Stutz, M. Despont, and M. A. Lantz, “Mass-balanced through-wafer electrostatic x/y-scanner for parallel probe data storage,” in *34th International Conference on Micro & Nano Engineering, Athens, Greece*, (Greece), p. 92, Ergo Publications, 2008.

[34] I. A. R. Center, “Racetrack memory - spintronics devices research.” `http://www.almaden.ibm.com/spinaps/research/sd/?racetrack`.

[35] S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y.-C. Chen, R. M. Shelby, M. Salinga, D. Krebs, S.-H. Chen, H.-L. Lung, and C. H. Lam, “Phase-change random access memory: A scalable technology,” *IBM Journal of Research and Development*, vol. 52, pp. 465–479, July/September 2008.

[36] TOSHIBA, Electronic Components, Inc., “NAND vs. NOR Flash Memory – Technology Overview.” `http://www.toshiba.com/taec/components/Generic/Memory_Resources/NANDvsNOR.pdf`.

[37] P. Pavan, R. Bez, P. Olivo, and E. Zanoni, “Flash memory cells-an overview,” *Proceedings of the IEEE*, vol. 85, pp. 1248–1271, August 1997.

[38] A. Birrell, M. Isard, C. Thacker, and T. Wobber, “A design for high-performance flash disks,” *SIGOPS Operating Systems Review*, vol. 41, pp. 88–93, April 2007.

[39] Y. Zhu, “An overview on MEMS-based storage, its research issues and open problems,” in *SNAPI ’04: Proceedings of the international workshop on Storage network architecture and parallel I/Os*, pp. 48–57, September 2004.

[40] J. L. Griffin, S. W. Schlosser, G. R. Ganger, and D. F. Nagle, “Modeling and performance of MEMS-based storage devices,” in *Proceedings of ACM SIGMETRICS 2000*, (Santa Clara, California, 17-21 June), pp. 56–65, 2000.

[41] H. S. Bucy, G. R. Ganger, and Contributors, “The DiskSim simulation environment version 3.0,” reference manual, School of Computer Science, Carnegie Mellon University, January 2003.

[42] T. Madhyastha and K. P. Yang, “Physical modeling of probe-based storage,” in *Proceedings of the 18th IEEE Symposium on Mass Storage Systems and Technologies*, pp. 207–224, April 2001.

[43] B. Hong and S. A. Brandt, “An analytical solution to a MEMS seek time model,” Tech. Rep. UCSC-CRL-02-31, Storage Systems Research Center, University of California, Santa Cruz, September 2002.

[44] M. Sivan-Zimet, “Workload based optimization of probe-based storage,” Master’s thesis, University of California, Santa Cruz, California, USA, September 2001.

[45] I. Dramaliev and T. Madhyastha, “Optimizing Probe-Based Storage,” in *Proceedings of the Second USENIX Conference on File and Storage Technologies (FAST’02)*, (San Francisco, CA, USA), pp. 103–114, 31 March – 2 April 2003.

[46] S. W. Schlosser, *Using MEMS-Based Storage Devices in Computer Systems*. PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, May 2004. Report Nr. CMU-PDL-04-104.

[47] H. Yu, D. Agrawal, and A. E. Abbadi, “Tabular placement of relational data on mems-based storage devices,” in *VLDB’2003: Proceedings of the 29th international conference on Very large data bases*, pp. 680–693, September 2003.

[48] H. Yu, D. Agrawal, and A. El Abbadi, “Declustering two-dimensional datasets over MEMS-based storage,” in *Advances in Database Technology – Proceedings of the 9th International Conference on Extending Database Technology (EDBT 2004)*, no. 2992 in Lecture Notes in Computer Science, (Heraklion, Crete, Greece, 14–18 March), pp. 495–512, Springer, 2004.

[49] J. L. Griffin, S. W. Schlosser, G. R. Ganger, and D. F. Nagle, “Operating System Management of MEMS-based Storage Devices,” in *Proceedings of the 4th Symposium on Operating Systems Design & Implementation*, (San Diego, California, 23-25 Oct.), pp. 227–242, October 2000.

[50] S. W. Schlosser and G. R. Ganger, “Mems-based storage devices and standard disk interfaces: A square peg in a round hole?,” in *FAST’04: Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, (Berkeley, CA, USA), pp. 87–100, USENIX Association, 2004.

[51] B. Hong, S. A. Brandt, D. D. E. Long, E. L. Miller, K. A. Glocer, and Z. N. J. Peterson, “Zone-based shortest positioning time first scheduling for MEMS-based storage devices,” in *Proceedings of the 11th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2003)*, (Orlando, FL), pp. 104–113, October 2003.

[52] Y. Lin, S. A. Brandt, D. D. E. Long, and E. L. Miller, “Power conservation strategies for mems-based storage devices,” in *Proceedings of the Tenth IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS 2002)*, pp. 53–62, October 2002.

[53] S. W. Schlosser, J. L. Griffin, D. F. Nagle, and G. R. Ganger, “Designing Computer Systems with MEMS-based Storage,” in *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 1–12, October 2000.

[54] F. Wang, B. Hong, S. A. Brandt, and D. D. E. Long, “Using MEMS-based storage to boost disk performance.,” in *MSST'05: 22nd IEEE Conference on Mass Storage Systems and Technologies*, pp. 202–209, April 2005.

[55] R. Rangaswami, Z. Dimitrijevic, E. Chang, and K. E. Schauser, “Mems-based disk buffer for streaming media servers,” *Data Engineering, 2003. Proceedings. 19th International Conference on*, pp. 619–630, March 2003.

[56] M. Uysal, A. Merchant, and G. A. Alvarez, “Using MEMS-Based Storage in Disk Arrays,” in *FAST '03: Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, (Berkeley, CA, USA), pp. 89–101, March 31 – April 2 2003.

[57] “LabVIEW.” `http://en.wikipedia.org/wiki/LabVIEW`. Accessed in February 2009.

[58] “OpenEmbedded.” `http://www.openembedded.org/`. Accessed in September 2007.

[59] “SanDisk Standard CompactFlash card.” `http://www.sandisk.com/OEM/ProductCatalog(1337)-CompactFlash_Memory_Card.aspx`. Accessed in April 2007.

[60] “The Familiar Project.” `http://familiar.handhelds.org/`. Accessed in February 2009.

[61] A. D. Brunelle, *Blktrace User Guide*, April 2006. `http://www.scribd.com/doc/2288714/blktrace-usage`.

[62] A. Sebastian, A. Pantazi, G. Cherubini, M. Lantz, H. Rothuizen, H. Pozidis, and E. Eleftheriou, “Towards faster data access: Seek operations in MEMS-based storage devices,” *Control Applications, 2006. CCA '06. IEEE International Conference on*, pp. 283–288, October 2006.

[63] E. Eleftheriou, P. Bächtold, G. Cherubini, C. Hagleitner, T. Loeliger, A. Pantazi, H. Pozidis, T. Albrecht, G. K. Binnig, M. Despont, U. Drechsler, U. Dürig, B. Gotsmann, D. Jubin, W. Häberle, M. A. Lantz, H. Rothuizen, R. Stutz, P. Vettiger, and D. Wiesmann, “A nanotechnology-based approach to data storage,” in *Proceedings of the 29th VLDB Conference*, (Berlin, Germany), 2003.

[64] B. Jacob, S. W. Ng, and D. T. Wang, *Memory Systems (Cache, DRAM, Disk)*. Morgan Kaufmann, 2008.

[65] "The third extended file system (ext3)." http://en.wikipedia.org/wiki/Ext3. Accessed in February 2009.

[66] "The IOzone Filesystem Benchmark." http://www.iozone.org/.

[67] S. Gurumurthi, *Power Management of Enterprise Storage Systems.* PhD thesis, The Pennsylvania State University, University Park, Pennsylvania, 2005.

[68] "MK4001MTD: 0.85" HDD 4 GB." http://sdd.toshiba.com/main.aspx?Path=StorageSolutions/0.85-inchHardDiskDrives/MK4001MTD. Accessed in April 2009.

[69] L. Benini, A. Bogliolo, and G. De Micheli, "A Survey of Design Techniques for System-Level Dynamic Power Management," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 8, pp. 299–316, June 2000.

[70] Y.-H. Lu, E.-Y. Chung, T. Šimunić, L. Benini, and G. De Micheli, "Quantitative comparison of power management algorithms," in *Proceedings of the conference on Design, automation and test in Europe (DATE)*, pp. 20–26, ACM Press, March 2000.

[71] S. McCarthy, M. Leis, and S. Byan, "Larger Disk Blocks or Not?," tech. rep., 2002.

[72] R. Berger, Y. Cheng, R. Forch, B. Gotsmann, J. Gutmann, T. Pakula, U. Rietzler, W. Schartl, M. Schmidt, A. Strack, J. Windeln, and H.-J. Butt, "Nanowear on polymer films of different architecture," *Langmuir*, vol. 23, pp. 3150–3156, March 2007.

[73] B. Bhushan, K. J. Kwak, and M. Palacio, "Nanotribology and nanomechanics of AFM probe-based data recording technology," *Journal of Physics: Condensed Matter*, vol. 20, p. 365207 (34pp), August 2008.

[74] M. Hinz, O. Marti, B. Gotsmann, M. A. Lantz, and U. Durig, "High resolution vacuum scanning thermal microscopy of HfO_2 and SiO_2," *Applied Physics Letters*, vol. 92, p. 043122, August 2008.

[75] H. Bhaskaran, A. Sebastian, and M. Despont, "Nanoscale PtSi tips for conducting probe technologies," *Nanotechnology, IEEE Transactions on*, vol. 8, pp. 128–131, January 2009.

[76] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *Journal of the ACM*, vol. 36, pp. 335–348, April 1989.

[77] E. Gal and S. Toledo, "Algorithms and data structures for flash memories," *ACM Computing Survey*, vol. 37, pp. 138–163, June 2005.

[78] "Barrier - a synchronization mechanism in parallel computing." `http://en.wikipedia.org/wiki/Barrier_(computer_science)`. Accessed in February 2009.

[79] L.-P. Chang, "On efficient wear leveling for large-scale flash-memory storage systems," in *SAC'07: Proceedings of the 2007 ACM symposium on Applied computing*, (New York, NY, USA), pp. 1126–1130, March 2007.

[80] J. Lorch and A. Smith, "Software strategies for portable computer energy management," *IEEE Personal Communications*, vol. 5, pp. 60–73, June 1998.

[81] Ö. Mesut, B. van den Brink, J. Blijlevens, E. Bos, and G. de Nijs, "Hard disk drive power management for multi-stream applications," in *International Workshop on Software Support for Portable Storage (IWSSPS)*, March 2005.

[82] "Hitachi Global Storage Technologies. OEM Hard Disk specifications for HTC424040F9AT00. Hitachi Travelstar C4K40 1.8-inch drive," November 2003. Revision (7).

[83] "SanDisk Extreme-III CompactFlash card." `http://www.sandisk.com/Products/ProductInfo.aspx?ID=1181`. Accessed in April 2007.

[84] J. W. Janzen, *TN-46-03 - Calculating Memory System Power for DDR*. Micron Technology, Inc., October 2003.

[85] P. M. Greenawalt, "Modeling power management for hard disks," in *MASCOTS'94: Proceedings of the Second International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems*, (Washington, DC, USA), pp. 62–66, IEEE Computer Society, 1994.

[86] "International Technology Roadmap for Semiconductors (ITRS) 2006 update," 2006. `http://www.itrs.net/Links/2006Update/2006UpdateFinal.htm`.

[87] T. Bisson, S. Brandt, and D. Long, "A hybrid disk-aware spin-down algorithm with I/O subsystem support," *Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE Internationa*, pp. 236–245, April 2007.

APPENDIX A

EXCERPTS FROM COLLECTED TRACES

A.1 I/O Trace

We collect I/O traces on our experimental platform at a point where I/O requests are dispatched to the storage device. In our study, the storage device is a CompactFlash (CF) card. The trace is then encoded in the DiskSim "ascii" format [41]. Figure A.1 shows an excerpt from the `scenarios` trace after conditioning. The trace is captured when formatting with the `ext3` file system and a block size of 4 KB.

Each line of the trace corresponds to one I/O request. A request is represented by a record of five fields. The first field is the arrival time of the request in seconds that is dispatched to a certain device specified by the second field. The address of the request, its size, and its operation ("1" for read and "0" for write) are given in fields 3 – 5, respectively. The last field is the identity number (ID) of the process responsible for the request.

A.2 Power Trace

We measure the voltage drop across a 1 Ω resistor to calculate the power dissipated by the component under question. We probe the power dissipated by the storage device and the PDA as a whole (including the storage device). The CF card has two VCCs, both of which we probe. Thus, each record of the power trace consists of four fields: (1) time stamp, (2) power due to the VCC of the PDA, (3) power due to the VCC 1, and (4) power due to the VCC 2 of the CF

Time	Device no.	Address	Size	Read/ Write	Process ID
0.000105	0	2657392	8	1	486
0.184202	0	2133512	8	1	507
0.235512	0	2133688	32	1	509
1.872268	0	2656144	8	1	542
4.328739	0	263056	8	0	256
4.489787	0	103728	184	0	256
4.633253	0	103912	8	0	256
5.984881	0	2656208	8	1	762
6.736644	0	2656064	16	1	801
6.779029	0	2655360	8	1	801
6.781702	0	2655432	8	1	801
6.904309	0	3441816	16	1	809
6.969250	0	3443560	64	1	812
6.985154	0	3443624	40	1	812
6.996180	0	3443664	8	1	812
7.011786	0	2655392	8	1	812
7.015291	0	2655400	8	1	812
7.021188	0	2655416	8	1	812
7.024833	0	2655408	8	1	812
7.066991	0	2655424	8	1	812
7.147231	0	361392	8	1	817
7.220469	0	2655896	8	1	820
7.387711	0	2130824	8	1	829
7.471849	0	3443368	8	1	833
8.139471	0	2137552	32	1	870
9.368797	0	103920	144	0	256
9.403790	0	104064	8	0	256
10.680199	0	2655368	8	1	1040

Figure A.1: The first 10 seconds from the `scenarios` trace when formatting with the `ext3` file system and a 4 KB block size. Time is given in seconds, and the request address and size are in a granularity of 512-byte sector. Read and write requests are encoded as 1 and 0, respectively. Time resolution is 1 µs.

card. Figure A.2 shows an excerpt.

A.3 Control Log

The control log is maintained by the logging machine, which stores the captured traces. We record in the log the time at which capturing the I/O trace

```
Time     PDA VCC   CF VCC1   CF VCC2
0.000    1.0802    0.0089    0.0103
0.001    1.0568    0.0088    0.0103
0.002    0.9680    0.0086    0.0098
0.003    1.0800    0.0092    0.0103
0.004    1.0520    0.0089    0.0103
0.005    1.0890    0.0095    0.0105
0.006    1.1095    0.0090    0.0105
0.007    1.1019    0.0093    0.0106
0.008    1.0933    0.0092    0.0103
0.009    1.0580    0.0093    0.0104
0.010    1.0905    0.0095    0.0103
0.011    1.1033    0.0088    0.0103
0.012    1.0850    0.0090    0.0104
0.013    1.1017    0.0092    0.0103
0.014    1.1164    0.0092    0.0105
0.015    1.1188    0.0090    0.0105
0.016    1.0975    0.0094    0.0102
0.017    1.0764    0.0094    0.0104
0.018    1.0666    0.0093    0.0106
0.019    1.0691    0.0093    0.0104
0.020    1.0598    0.0091    0.0105
```

Figure A.2: An excerpt of 20 micro seconds from the power trace that corresponds to the `scenarios` trace when formatting with the `ext3` file system and a 4 KB block size. Time is given in micro seconds, and power is given in Watts. The resolution of voltage is 6.1 μV.

and the power trace starts. This allows us to calculate the difference in timing due to, for example, network delay between the I/O trace and the power trace. Further, we capture the starting time of each session. Recall that a trace is captured by running several sessions in sequence on the application level. Capturing the session starting time allows us to measure the energy consumption and to simulate for a specific session. As a result, we can attribute requests to applications and reason with their behavior as used in Section 4.2.3. Figure A.3 shows an example control log.

A.4 Statistics of the I/O Traces

We present the statistics of our three traces (`scenarios`, `multimedia`, and `iozone`) in Figures A.4a–A.6c. We show the distribution of the seek distance, the request size, and the inter-arrival time for each trace.

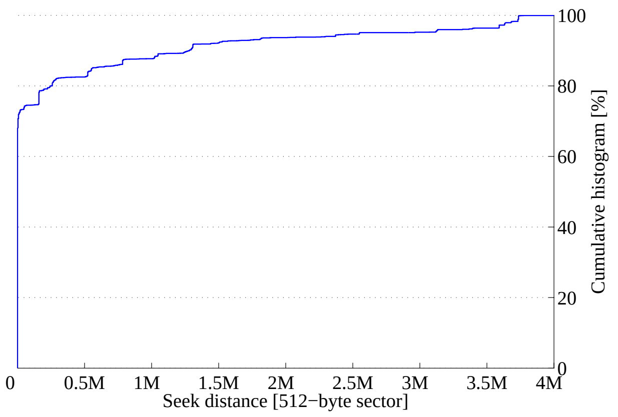
```
start,0.0,h2200/sandisk sdcfb-2048/ext3/61552kb/noop/00/ # at Mon Feb 05
 10:50:27 CET 2007 (1170669027.49985)
startpower,4.785097,h2200/sandisk sdcfb-2048/ext3/61552kb/noop/00/power
 # time-base offset!
stamp,0.0,initializing
starttrace,5.027494,h2200/sandisk sdcfb-2048/ext3/61552kb/noop/00/trace
 # time-base offset!
stamp,6.428285,booting
stamp,52.461748,starting-opie
stamp,79.297711,start-applications-sequentially
stamp,126.677935,playing-mp3
stamp,276.148661,writing-1MB-file-in-16KBs-indirectly
stamp,288.381833,writing-1MB-file-in-64KBs-indirectly
stamp,300.632934,reading-1M-stream-@-32Kbps-in-4KBs-indirectly
stamp,571.085983,reading-1M-stream-@-128Kbps-in-16KBs-indirectly
stamp,647.356769,reading-1M-stream-@-384Kbps-in-48KBs-indirectly
stamp,681.217931,writing-1M-stream-@-32Kbps-in-4KBs-indirectly
stamp,951.66178,writing-1M-stream-@-128Kbps-in-16KBs-indirectly
stamp,1027.990795,writing-1M-stream-@-384Kbps-in-48KBs-indirectly
stamp,1062.042972,removing-dump-files
stamp,1073.834049,copy-small-files
stamp,1086.223096,copy-5.5MB-file
stamp,1102.834319,start-applications-simultaneously
stamp,1145.694436,endSession
endpower,1145.694555
```

Figure A.3: An example control log of the `scenarios` trace when formatting with the `ext3` file system and a 4 KB block size. It shows how the tracing and probing are started first and then a series of sessions are executed.

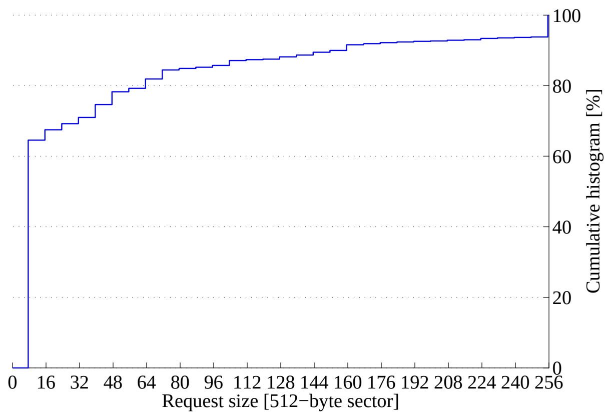
Form the figures we observe that a large percentage of the seek distances is below 500 sectors for the three traces. All the three requests exhibit seeks of large distances that span at least two-third of the address space.

The majority of requests are 4 KB in size. These requests are mainly due to the file system and system daemons. For all traces, we observe that 90% of the requests are smaller than or equal to 16 KB.

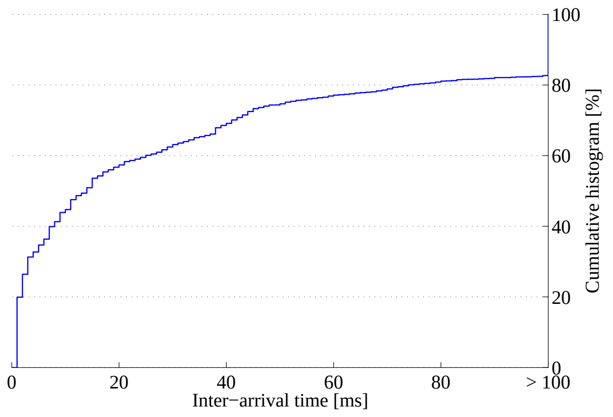
The three traces differ largely in the inter-arrival time. Since the IOzone benchmark stresses the device with consecutive requests, the `iozone` trace exhibits small inter-arrival times, unlike the other two traces. The multimedia trace exhibits large inter-arrival times, because of the pre-fetching applied in streaming environments.



(a) Seek distance

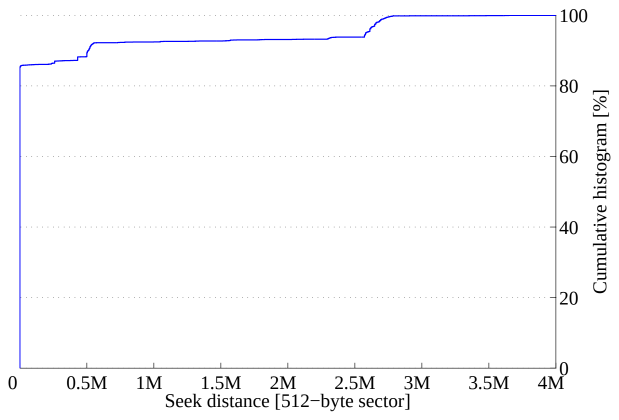


(b) Request size

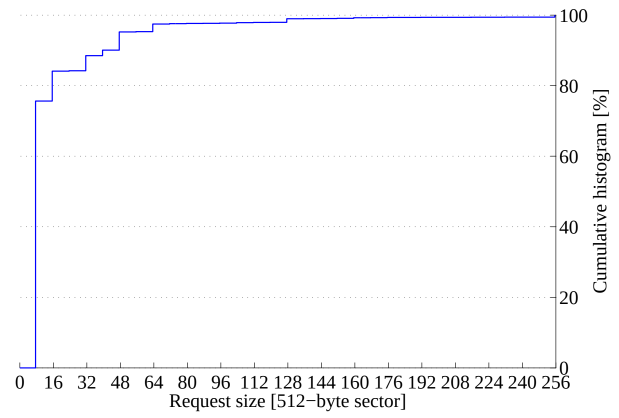


(c) Inter-arrival time

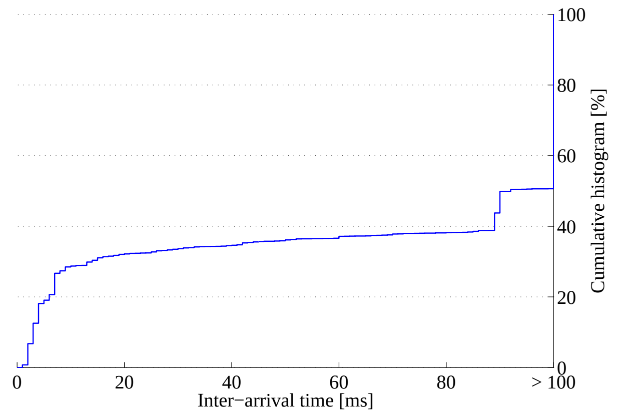
Figure A.4: Cumulative histogram of the seek distance, request size, and the inter-arrival time of I/O requests of the `scenarios` trace.



(a) Seek distance

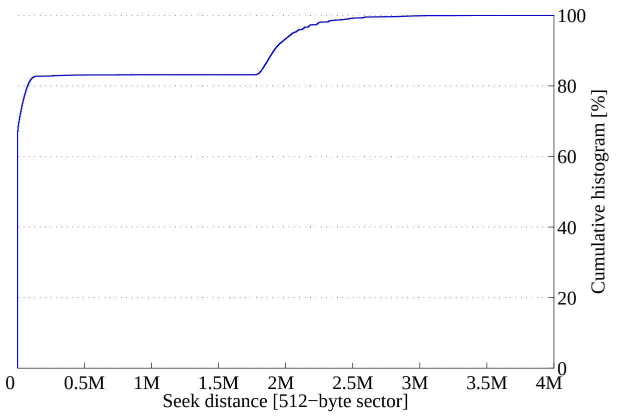


(b) Request size

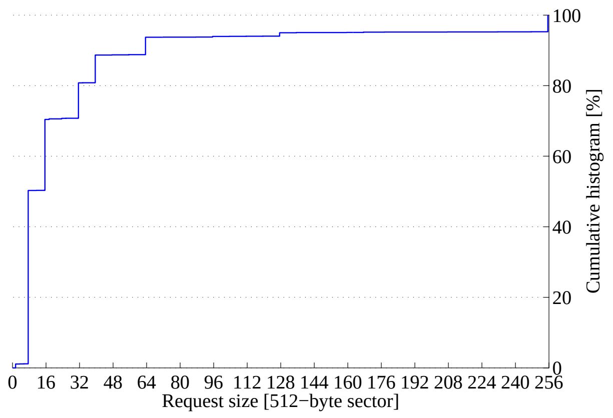


(c) Inter-arrival time

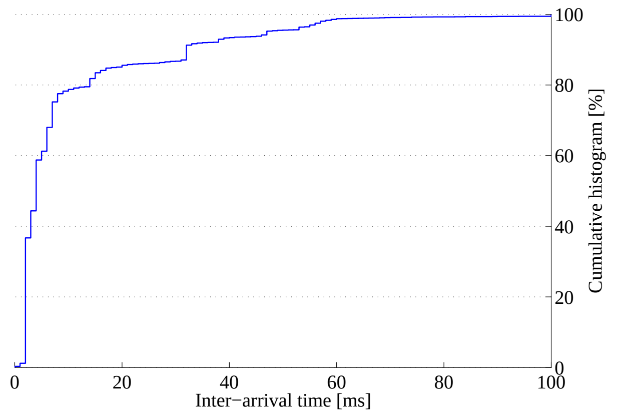
Figure A.5: Cumulative histogram of the seek distance, request size, and the inter-arrival time of I/O requests of the `multimedia` trace.



(a) Seek distance



(b) Request size



(c) Inter-arrival time

Figure A.6: Cumulative histogram of the seek distance, request size, and the inter-arrival time of I/O requests of the `iozone` trace.

APPENDIX B

Modeling Two Shutdown Policies

Shutdown is the operation in which a MEMS-based storage device prepares for going into the inactivity state. In shutdown mode, the device drives the media sled to its resting position, at which the sled dissipates no power. The resting position is the center of the probe field, and is conventionally taken as the $(0,0)$ coordinates.

The sled can be driven to the center position in two ways: The first is similar to seeking from the current position to the center position except that the sled stops on X as well as Y. Like seek operation, this shutdown policy uses the actuators to accelerate and the decelerate the sled as Figure B.1a shows. We call it the **performance-efficient shutdown policy**.

Another way of shutting down is to exploit only the potential energy stored in the springs to accelerate the sled toward the center. This policy uses the actuators only for deceleration to stop the sled at the center as shown in Figure B.1b. Therefore, it consumes less energy than the previous policy at the cost of longer shutdown time. we call it the **energy-efficient shutdown policy**.

B.1 Modeling

We use the bang-bang model presented in Section 3.2.2 to estimate the shutdown time for the performance-efficient shutdown policy. For this policy, the analytical derivation of the shutdown time is similar to that of the seek time provided by Hong *et al.* [43] except for the full stop on the Y direction. Because of this similarity, we only provide the final formulas and point out the

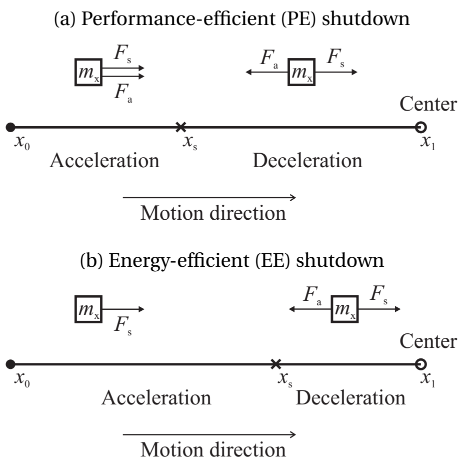


Figure B.1: A sketch of the sled motion toward the center when shutting down with the performance-efficient and energy-efficient shutdown policies

modifications in Hong *et al.'s* derivation.

The energy-efficient policy involves no external forces, and thus has different acceleration dynamics. We provide a complete derivation of the acceleration and deceleration parts of the shutdown operation when adopting this policy. We adopt the bang-bang model, whereby the actuators apply the maximum force to decelerate the sled. We list the model parameters in Table C.1.

According to Newton's second law, the motion of the sled in the acceleration phase is described in general as follows:

$$m_{\mathrm{x}} \cdot \ddot{x} = m_{\mathrm{x}} \cdot a_{\mathrm{x}} - k_{\mathrm{x}} \cdot x,$$

the first term of the equation becomes zero for the energy-efficient policy. For the deceleration phase is described in general as follows:

$$m_{\mathrm{x}} \cdot \ddot{x} = -m_{\mathrm{x}} \cdot a_{\mathrm{x}} - k_{\mathrm{x}} \cdot x.$$

B.2 The Energy-Efficient Policy

In MEMS-based storage devices the media sled is suspended by springs across the probe array. If no external force is applied, the springs pull the medium back to the center.

To save investing external energy (i.e., from the host system), the energy-efficient shutdown policy exploits of the potential energy stored in the springs to accelerate the sled toward the center. To bring the sled stationary at the

Table B.1: List of the parameters of the analytical models of the shutdown time

Parameter	Symbol	Value	Unit
actuator resistor	R_{coil}	8.4	Ω
maximum current	$i_{\max}$	0.2	A
sled scan speed	v_{a}	0.001	m/s
X spring constant	k_{x}	104.0	N/m
Y spring constant	k_{y}	91.0	N/m
sled mass on X	m_{x}	0.000102	kg
sled mass on Y	m_{y}	0.000082	kg
maximum acceleration on X	a_{x}	51.17	m/s^2
maximum acceleration on Y	a_{y}	55.73	m/s^2
acceleration due to springs[a]	a_{s}	–	m/s^2

[a] Acceleration due to springs depends on the displacement of the spring.

center, we determine a *switching point*, on each direction, at which a counter force is applied by the actuators to decelerate the sled.

Thus, when deploying the energy-efficient policy, energy (from the host system) is consumed only during deceleration and not during acceleration. In the following, we derive the analytical model for the motion along the X and Y directions separately; we start with X followed by Y.

B.2.1 Motion along X

We assume that the sled is at position x_0 and moves to the center $x_1 = 0$. Deceleration starts at a switching point, x_{s}. The switching point is determined using the energy conservation equation as follows:

$$\begin{aligned}\sum E = 0 \quad &\Rightarrow \quad \frac{k_{\mathrm{x}}}{2} x_0^2 - (x_{\mathrm{s}} - x_1) \cdot F = 0 \\ &\Rightarrow \quad x_{\mathrm{s}} = \frac{k_{\mathrm{x}}}{2F} \cdot x_0^2, \end{aligned} \tag{B.1}$$

where F is the counter force applied by the actuators to stop the the sled at the center. The spring motion toward the center is split in two phases; the acceleration and the deceleration phase. The acceleration phase is between x_0 and x_{s}, whereas the deceleration phase is between x_{s} and x_1 (as depicted in Figure B.1a).

Acceleration phase According to Newton's second law, the motion of the sled in the acceleration phase is described as follows:

$$\begin{aligned} m_\mathrm{x} \cdot \ddot{x} &= -k_\mathrm{x} \cdot x \\ \Rightarrow \quad \ddot{x} &= -\frac{k_\mathrm{x}}{m_\mathrm{x}} \cdot x. \end{aligned}$$

This is an ordinary differential equation, whose general solution has the form:

$$x(t) = C_1 \cdot \cos(\sqrt{\frac{k_\mathrm{x}}{m_\mathrm{x}}} \cdot t) + C_2 \cdot \sin(\sqrt{\frac{k_\mathrm{x}}{m_\mathrm{x}}} \cdot t)$$

From the boundary conditions, $x(t=0) = x_0$ and $\dot{x}(t=0) = 0$, we get $C_1 = x_0$ and $C_2 = 0$, respectively. Thus:

$$x(t) = x_0 \cdot \cos(\sqrt{\frac{k_\mathrm{x}}{m_\mathrm{x}}} \cdot t) \tag{B.2}$$

Rearranging Equation (B.3), we get the acceleration time, $t_\mathrm{a,x}$, between x_0 and x_s:

$$t_\mathrm{a,x} = \sqrt{\frac{m_\mathrm{x}}{k_\mathrm{x}}} \cdot \arccos(\frac{x_\mathrm{s}}{x_0}). \tag{B.3}$$

Deceleration phase In the deceleration phase the actuators exert a force to stop the sled at the center, which works against the spring force. The motion is described as follows:

$$\begin{aligned} m_\mathrm{x} \cdot \ddot{x} &= -m_\mathrm{x} \cdot a_\mathrm{x} - k_\mathrm{x} \cdot x \\ \Rightarrow \quad \ddot{x} &= -a_\mathrm{x} - \frac{k_\mathrm{x}}{m_\mathrm{x}} \cdot x \end{aligned}$$

Here a_x is the acceleration due to the actuators, and the minus sign before it indicates that it opposes the motion direction of the sled. This motion equation has the following general solution:

$$x(t) = C_1 \cdot \cos(\sqrt{\frac{k_\mathrm{x}}{m_\mathrm{x}}} \cdot t) + C_2 \cdot \sin(\sqrt{\frac{k_\mathrm{x}}{m_\mathrm{x}}} \cdot t) - \frac{m_\mathrm{x} \cdot a_\mathrm{x}}{k_\mathrm{x}}$$

Since $x(t=0) = x_\mathrm{s}$, we get:

$$C_1 = x_\mathrm{s} + \frac{m_\mathrm{x} \cdot a_\mathrm{x}}{k_\mathrm{x}}.$$

Two boundary conditions exist: $x(t = t_\mathrm{d,x}) = x_1$ and $\dot{x}(t = t_\mathrm{d,x}) = 0$. We use them to calculate the deceleration time, $t_\mathrm{d,x}$, and then C_2:

$$t_\mathrm{d,x} = \sqrt{\frac{m_\mathrm{x}}{k_\mathrm{x}}} \cdot \arccos(\frac{x_\mathrm{s} + \frac{m_\mathrm{x} \cdot a_\mathrm{x}}{k_\mathrm{x}}}{x_1 + \frac{m_\mathrm{x} \cdot a_\mathrm{x}}{k_\mathrm{x}}}), \tag{B.4}$$

and

$$C_2 = C_1 \cdot \tan(\sqrt{\frac{k_x}{m_x}} \cdot t_{x_d}).$$

The solution given in Equation (B.4) holds for $x_s < x_1$. When $x_s > x_1$, as in our case, since $x_1 = 0$ (the center), we multiply every x by -1, so that $-x_0 < -x_1$.

Total shutdown time along X From Equation (B.3) and Equation (B.4), we calculate the total shutdown time along X is:

$$t_{\text{shutdown,x}} = t_{\text{a,x}} + t_{\text{d,x}}. \tag{B.5}$$

B.2.2 Motion along Y

We assume that the sled is at position y_0 and moves to the center $y_1 = 0$. Deceleration starts at a switching point, y_s. Unlike the X direction, on which the sled stays still, the sled moves along Y to access data. Here, we distinguish two states of the sled when a decision is made to shut it down: (1) the sled moves along Y in the direction of the center (inward), or (2) the sled moves toward the borders (outward).

If the sled moves inward, then an additional initial speed should be considered, namely its data access speed (v_a). On the other hand, if it moves outward, it should brake first and then accelerates toward the center, starting at a velocity of zero. Thus, when moving outward the sled motion is identical to that discussed for X, and thus the shutdown time is calculated similarly to that on X, while adding a brake time:

$$t_{\text{brake}} = \frac{v_a}{a_s}. \tag{B.6}$$

In the following, we study the first case, where the sled accelerates toward the center with an initial velocity v_a. The switching point (y_s) is:

$$\frac{k_y}{2} \cdot y_0^2 - (y_s - y_1) \cdot F - \frac{m_y}{2} \cdot v_a^2 = 0$$
$$\Rightarrow \quad y_s = \frac{k_y \cdot y_0^2 - m_y \cdot v_a^2}{2 \cdot F} \tag{B.7}$$

Here F is the actuation force exerted by the actuators between y_s and $y_1 = 0$. The motion along Y is also split into an acceleration and deceleration phase.

Acceleration phase The motion equation for this phase is:

$$\ddot{y} = -\frac{k_y}{m_y} \cdot y$$

The solution for this equation has the form:

$$y(t) = C_1 \cdot \cos(\sqrt{\frac{k_y}{m_y}} \cdot t) + C_2 \cdot \sin(\sqrt{\frac{k_y}{m_y}} \cdot t)\,.$$

Since $y(t = 0) = y_0$ and $\dot{y}(t = 0) = v_a \cdot$, we get $C_1 = y_0$ and $C_2 = v_a \cdot$, respectively. Thus:

$$y(t) = y_0 \cdot \cos(\sqrt{\frac{k_y}{m_y}} \cdot t) + v_a \cdot \sin(\sqrt{\frac{k_y}{m_y}} \cdot t) \tag{B.8}$$

Rearranging Equation (B.8), we get the acceleration time along Y, $t_{a,y}$:

$$t_{a,y} = \sqrt{\frac{m_y}{k_y}} \cdot (\arcsin(\frac{y_s}{\sqrt{y_0^2 + (v_a \cdot \sqrt{\frac{m_y}{k_y}})^2}}) - \arcsin(\frac{y_0}{\sqrt{y_0^2 + (v_a \cdot \sqrt{\frac{m_y}{k_y}})^2}}))\,. \tag{B.9}$$

We multiply every y by -1, so that $-y_s < -y_1$.

Deceleration phase The deceleration phase along Y has the same initial (it starts at y_s) and boundary conditions ($y(t = t_{d,y}) = y_1$ and $\dot{y}(t = t_{d,y}) = 0$) like that along X. Thus, the deceleration time is:

$$t_{d,y} = \sqrt{\frac{m_y}{k_y}} \cdot \arccos(\frac{y_s + \frac{m_y \cdot a_y}{k_y}}{x_1 + \frac{m_y \cdot a_y}{k_y}})\,. \tag{B.10}$$

We multiply every y by -1 so that $-y_s < -y_1$.

Total shutdown time along Y The total shutdown time along Y is:

$$t_{\text{shutdown},y} = t_{a,y} + t_{d,y} + \begin{cases} t_{\text{brake}} & ;\text{ if outward} \\ 0 & ;\text{ if inward} \end{cases} \tag{B.11}$$

If the sled is moving outward at the shutdown decision (case 2), $t_{a,y}$ and $t_{d,y}$ are calculated using Equations (B.3) and (B.4), respectively, after replacing every parameter of X with its Y counterpart. Otherwise, if the sled is moving inward (case 1), we use Equations (B.9) respectively (B.10) instead. Also, y_s is calculated using Equation (B.1) or (B.7), respectively.

B.3 The Performance-Efficient Policy

The performance-efficient policy enables the sled to reach the center in the shortest time possible. Therefore, we use the actuators for acceleration and deceleration, consuming external energy in both phases. When deploying the performance-efficient policy, the sled acts as if it seeks from the current position to the center position. Unlike in seeking, in shutdown the sled has to stop at the center on X as well as Y. Hong *et al.* [43] devise an analytical bang-bang model for the seek time. The model of the motion along X holds completely for the shutdown time along X; the acceleration time is:

$$t_{\text{a,x}} = \sqrt{\frac{m_{\text{x}}}{k_{\text{x}}}} \cdot \arccos\Big(\frac{x_{\text{s}} - \frac{m_{\text{x}} \cdot a_{\text{x}}}{k_{\text{x}}}}{x_0 - \frac{m_{\text{x}} \cdot a_{\text{x}}}{k_{\text{x}}}}\Big)\,, \tag{B.12}$$

and the deceleration time is:

$$t_{\text{d,x}} = \sqrt{\frac{m_{\text{x}}}{k_{\text{x}}}} \cdot \arccos\Big(\frac{x_{\text{s}} + \frac{m_{\text{x}} \cdot a_{\text{x}}}{k_{\text{x}}}}{x_1 + \frac{m_{\text{x}} \cdot a_{\text{x}}}{k_{\text{x}}}}\Big)\,. \tag{B.13}$$

Unlike the X seek model, the Y seek model needs modification to account for the stop on Y at the center. The acceleration time along Y becomes:

$$\begin{aligned} t_{\text{a,y}} = {} & \sqrt{\frac{m_{\text{y}}}{k_{\text{y}}}} \cdot \arcsin\Bigg(\frac{y_{\text{s}} - \frac{m_{\text{y}} \cdot a_{\text{y}}}{k_{\text{y}}}}{\sqrt{(y_0 - \frac{m_{\text{y}} \cdot a_{\text{y}}}{k_{\text{y}}})^2 + (v_{\text{a}} \cdot \sqrt{\frac{m_{\text{y}}}{k_{\text{y}}}})^2}}\Bigg) \\ & - \sqrt{\frac{m_{\text{y}}}{k_{\text{y}}}} \cdot \arcsin\Bigg(\frac{y_0 - \frac{m_{\text{y}} \cdot a_{\text{y}}}{k_{\text{y}}}}{\sqrt{(y_0 - \frac{m_{\text{y}} \cdot a_{\text{y}}}{k_{\text{y}}})^2 + (v_{\text{a}} \cdot \sqrt{\frac{m_{\text{y}}}{k_{\text{y}}}})^2}}\Bigg)\,, \end{aligned} \tag{B.14}$$

and the deceleration time is:

$$t_{\text{d,y}} = \sqrt{\frac{m_{\text{y}}}{k_{\text{y}}}} \cdot \arccos\Big(\frac{y_{\text{s}} + \frac{m_{\text{y}} \cdot a_{\text{y}}}{k_{\text{y}}}}{y_1 + \frac{m_{\text{y}} \cdot a_{\text{y}}}{k_{\text{y}}}}\Big)\,. \tag{B.15}$$

The solution given in the previous equations holds for $x_{\text{s}} < x_1$. When $x_{\text{s}} > x_1$, as in our case, since $x_1 = 0$ (the center), we multiply every x by -1, so that $-x_0 < -x_1$. The same applies for y.

B.3.1 Shutdown time and energy

The motions along X and Y directions are independent, because of the independent actuators and their structure. Thus, the time to shut down is the maximum of the shutdown times along X and Y:

$$t_{\text{shutdown}} = \max(t_{\text{shutdown,x}}, t_{\text{shutdown,y}})\,. \tag{B.16}$$

Remember that no external energy is consumed to brake the sled if it moves outward. Also, no external energy is consumed to accelerate the sled; we use the potential energy stored in the springs instead. Thus, the total energy is the sum of the energy consumed only to decelerate the sled along X and along Y.

We drive the actuators with the maximum allowed current (thus the maximum power) to shorten the deceleration time. Thus, the total energy is:

$$E_{\text{shutdown}} = i_{\text{max}}^2 \cdot R_{\text{coil}} \cdot (t_{\text{d,x}} + t_{\text{d,y}}) . \tag{B.17}$$

Again, we use Equation (B.4) to calculate $t_{\text{d,y}}$ if the sled is moving outward. If it is moving inward, we use Equation (B.10).

B.4 Implementation in DiskSim

The previous MEMS model of DiskSim assumes an instantaneous shutdown time of the media sled in MEMS-based storage devices. As a result, it assumes that the shutdown operation consumes no energy. Since MEMS-based storage devices shut down very often due to aggressive power management, an accurate modeling of the shutdown time and energy becomes necessary.

We, therefore, implement the two shutdown policies in the DiskSim MEMS model. Simulations can be run with either policy by adjusting the shutdown-policy parameter. For better modeling of a real MEMS-based storage device, we keep track of the position of the sled during shutdown. As a result, if a new request arrives during shutdown, we interrupt the shutdown operation at its current position and satisfy the request.

APPENDIX C

MODELING STREAMING ARCHITECTURES

The objective of our study is to build an energy-efficient streaming architecture, thus our focus is to minimize energy consumption. Next, we dissect the total energy consumed by a storage device/memory to find out the individual energy components.

C.1 Energy Components

Although storage technologies vary in the ways they consume energy, we can still develop a generic template to model the energy components for each particular technology. A storage device consumes two types of energy: **static** and **dynamic** energy. Static energy is consumed to retain the content of a store, such as the refresh energy of a DRAM. Static energy is consumed as long as the device is powered on and contains useful data. The amount of consumed energy is device dependent but predominantly determined by its capacity. Dynamic energy, on the other hand, represents the energy consumed by each activity of the device associated with accessing data. Figure C.1 shows a generic power-state machine with five states: active (read/write), idle, shutdown, standby, and startup. The dynamic energy is the sum of the energy consumed in each state.

For each state, Figure C.1 specifies the power dissipation (P_x) and the duration t_x of the respective state. The period for the active (read/write) and standby states, t_{RW} and t_{sb}, respectively, are determined by the workload. The period for the idle and shutdown states have predetermined and fixed lengths

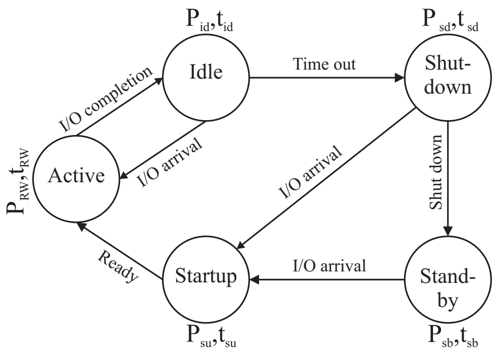


Figure C.1: A power-state machine of a storage device or memory. It shows per state the dynamically consumed energy. P_{RW} and t_{RW} refer to the read/write active power and time respectively.

after which a state transfer occurs. Finally, the period of the startup state, t_{su}, specifies the time involved for the device to become ready, which has a fixed device-dependent part and a dynamic workload dependent part. Note that an I/O arrival triggers a direct transition to the active state.

To reduce energy, we should reduce the static and/or the dynamic energy. Static energy is reduced by, among others, decreasing the device capacity. Reducing the dynamic energy requires: (1) limitation of the number of state transitions from standby to active, (2) avoidance of the idle state, and (3) extension of the standby periods.

As an example, assume a storage device that implements the power-state machine of Figure C.1. Deploying a buffer for such a device helps to reduce the dynamic energy of the device. For instance, if we stream 10 MB of data from the storage device, deploying a buffer with a capacity of 2 MB, then the power-state machine triggers five active-to-standby transitions, whereas if we deploy a 5 MB buffer, just two of these transitions are triggered. Further, fewer transitions yield less transition time, which prolongs the period that the device is in standby state. Thus, an appropriately sized (small) buffer saves energy.

C.2 Minimizing Energy

Minimizing the energy consumption of the DISk-Based Architecture (DISBA) requires a reduction of the disk dynamic energy and the DRAM static energy. To reduce the disk dynamic energy, we should increase its buffer, namely the DRAM, whereas to reduce the DRAM static energy its capacity should be reduced. Thus, we must find a DRAM capacity that minimizes the total energy consumption of the disk and the DRAM.

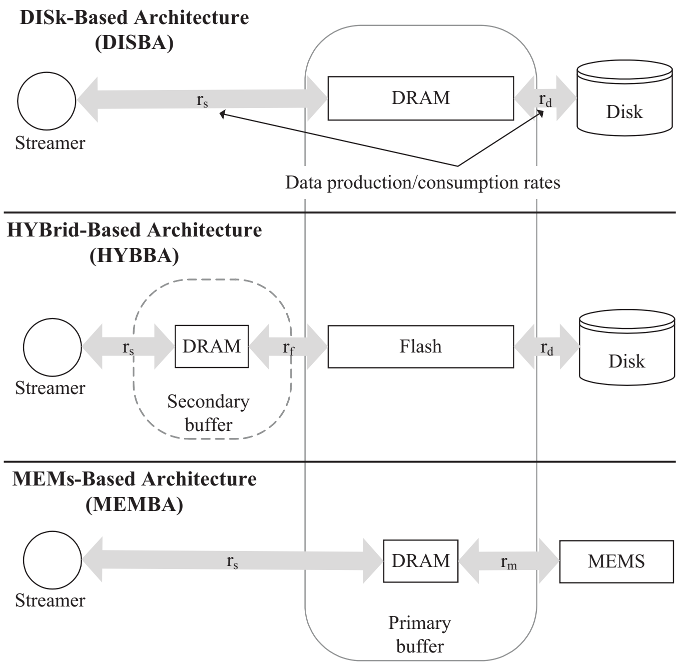

DISk-Based Architecture (DISBA)
Streamer
r_s
DRAM
r_d
Disk
Data production/consumption rates
HYBrid-Based Architecture (HYBBA)
Streamer
r_s
DRAM
r_f
Secondary buffer
Flash
r_d
Disk
MEMs-Based Architecture (MEMBA)
Streamer
r_s
DRAM
r_m
MEMS
Primary buffer

Figure C.2: Block diagram of DISBA (*top*) that exists today, HYBBA (*middle*) that could be implemented today, and MEMBA (*bottom*) that can be implemented once MEMS-based storage becomes available.

Minimizing the energy consumption of the HYBrid-storage–Based Architecture (HYBBA) requires reducing the disk dynamic energy, the flash dynamic energy, and the DRAM static energy. Reducing the disk energy does not conflict with reducing the flash energy, since the latter has no static energy. However, reducing the flash dynamic energy requires increasing the DRAM capacity, which increases DRAM energy. Because flash has no startup and shutdown overheads, the DRAM capacity is small and so is its energy, resolving the conflict.

Minimizing the energy consumption of MEMs-storage–Based Architecture (MEMBA) is achieved as in DISBA, since MEMS-based storage devices are mechanical like disk drives. The only difference is that a MEMS-based storage device has small overheads compared to disk drives (Section 4.1.1 on page 48). Further, a MEMS-based storage device has no startup overhead, so that frequent refills are possible. As a result, a small DRAM capacity is sufficient as a streaming buffer in MEMBA. A block diagram of the three architectures is shown in Figure C.2.

In summary, the objective is to minimize the total energy by (a) maximizing the capacity of the primary buffer in each architecture to reduce the energy consumed by the backing store, and (b) minimizing the capacity of the secondary buffer in HYBBA to save on the DRAM retention energy. This optimization problem has three constraints: (1) the capacity of the buffers should be larger than or equal to the capacity of the real-time buffer to guarantee smooth streaming, (2) the throughput of each buffer should be larger than that of the component right below it in the hierarchy, so that it does become a performance bottleneck, and (3) the buffer capacities are limited by the budget the designer is willing to spend.

To satisfy the first constraint, we derive the real-time buffer in the following section to ensure that the buffer capacities are sufficiently large. For the second constraint, we deploy as many physical modules as necessary to achieve the demanded throughput, and operate them in parallel, so that bottlenecks are avoided. The third constraint is, however, relaxed in order to investigate the full potential of the energy saving as a function of the buffer capacity. In fact, this relaxation leads us to two key findings. Firstly increasing the buffer capacity beyond a certain point leads to a larger increase in the system lifetime than in energy saving (Section 7.7). Secondly, a small buffer size is sufficient to achieve large energy savings for a reasonably long lifetime at negligible cost.

C.3 Buffer Capacities

Recall that the disk drive is the backing store in DISBA and HYBBA, whereas in MEMBA the MEMS-based storage device is the backing store. Since a MEMS-based storage device is mechanical and thus resembles a disk drive at an abstract level, we derive the models based on the disk drive. Modeling MEMBA can be achieved by substituting the parameters of MEMS-based storage in the disk-based models where applicable. The inapplicable parameters are set to zero. For the sake of brevity we consider throughout our study streaming from the disk drive. Nonetheless, our analysis applies equally to streaming to the disk drive or a mix of read/write streaming.

In this section, we analytically derive the capacities of the primary and secondary buffers. Since various types of latency are incurred when accessing data from a storage device, continuous streaming should be guaranteed. To guarantee continuous streaming, each of the primary and secondary buffers should be larger than a minimum buffer capacity, called the real-time buffer. We derive the real-time buffer capacity first. Table C.1 lists the key parameters of the models devised in Sections C.3–C.5.

C.3.1 The Real-Time Buffer Capacity

In streaming environments, we deal with (soft) real-time applications where throughput should be guaranteed and deadlines should be met to prevent

Table C.1: Key parameters of the analytical models in Sections C.3–C.5

Parameter	Description
	input
r_f	Flash throughput
r_d	Disk throughput
r_s	stream bit rate
	intermediate
t_bs	best-effort slack
T_d	disk refill cycle without best-effort service
T'_d	refill cycle with best-effort
B_be	capacity of the break-even buffer of the disk
$B_\text{rt-f}$	capacity of the real-time buffer of the Flash
	output
B_pm	capacity of the primary buffer
B_sc	capacity of the secondary buffer in HYBBA
	model parameters
α	scaling factor of the primary buffer ($B_\text{pm} = \alpha \cdot B_\text{be}$)
β	scaling factor of the secondary buffer ($B_\text{sc} = \beta \cdot B_\text{rt-f}$)
γ	best-effort slack percentage of T_d ($t_\text{bs} = \gamma \cdot T_\text{d}$)

degradation in quality (due to frame dropping) and/or loss of streaming data (due to buffer overflow). The throughput requirement is guaranteed by building the system out of components that have sufficient processing speed and communication bandwidth. The deadline requirement, however, arises when dealing with resources that incur latency to satisfy requests; like waiting to position the disk head over the right data track before data can be transferred from the disk. This requirement can be met by deploying a buffer that can stream data during the waiting time. The minimum buffer capacity to prevent under-run of consumed data is the *real-time buffer* capacity B_rt.

Assume a given storage device that is capable of reading data at a sustained throughput of r, see Figure C.3a. Every request incurs a latency l from the storage device as well as other resources such as the I/O bus. To sustain real-time streaming at a throughput r_s from the storage device, a real-time buffer of capacity B_rt is needed. The real-time buffer is a leaky bucket, which is filled at a rate $r - r_\text{s}$ (Figure C.3b), and which should sustain data during disk unavail-

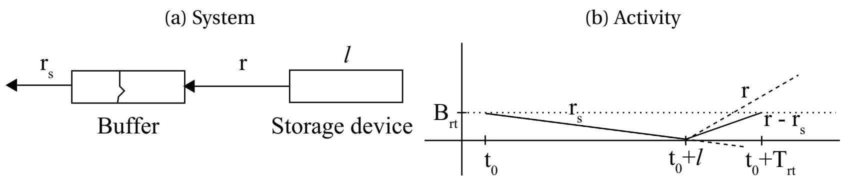


Figure C.3: Leaky bucket streaming system and its activity diagram

ability for l time units. Thus, its minimum capacity (B_{rt}) is:

$$B_{\mathrm{rt}} = l \cdot r_{\mathrm{s}} \tag{C.1}$$

This buffer should be refilled periodically. The cycle period, T_{rt}, is the sum of the incurred latency and the actual refill time:

$$T_{\mathrm{rt}} = \frac{B_{\mathrm{rt}}}{r_{\mathrm{s}}} + \frac{B_{\mathrm{rt}}}{r - r_{\mathrm{s}}} = \frac{l \cdot r}{r - r_{\mathrm{s}}}$$

Here, the throughput requirement is $r > r_{\mathrm{s}}$.

C.3.2 The Primary Buffer Capacity

The buffer that communicates directly with the backing store is referred to as the primary buffer. Streaming workloads have predictable data access patterns, so that their data can be fetched ahead into the primary buffer (See Figure 7.2), and consequently the disk can go into standby state to save energy. However, if the disk goes standby (still dissipating P_{sb} [J/s]), additional energy is consumed every cycle in the shutdown (dissipating P_{sd} [J/s]) state and the startup state (dissipating P_{su} [J/s]), which includes seeking in our model. To save energy using standby, this overhead must at least be compensated for, or the disk could be left spinning in idle mode (dissipating P_{id} [J/s]). Compensation of shutdown and startup overheads boils down to requiring a minimum standby period, t_{sb}, according to:

$$P_{\mathrm{id}} \cdot t_{\mathrm{id}} \geq t_{\mathrm{sb}} \cdot P_{\mathrm{sb}} + E_{\mathrm{oh}}, \tag{C.2}$$

where

$$\begin{aligned} t_{\mathrm{id}} &= t_{\mathrm{sb}} + t_{\mathrm{oh}} \\ t_{\mathrm{oh}} &= t_{\mathrm{su}} + t_{\mathrm{sd}} \\ E_{\mathrm{oh}} &= t_{\mathrm{su}} \cdot P_{\mathrm{su}} + t_{\mathrm{sd}} \cdot P_{\mathrm{sd}} \end{aligned}$$

The meaning of the parameters are shown in Figure C.1. Figure 7.2 shows graphically that the standby period should be sufficiently long to make area "b" larger than area "a". The idle period (t_{id}) that balances Inequality (C.2) is called the break-even period, t_{be}:

$$t_{\text{be}} = \frac{E_{\text{oh}} - t_{\text{oh}} \cdot P_{\text{sb}}}{P_{\text{id}} - P_{\text{sb}}},$$

and the corresponding *break-even buffer capacity* (B_{be}) is:

$$B_{\text{be}} = t_{\text{be}} \cdot r_{\text{s}}.$$

If the primary buffer has the capacity of the break-even buffer, the disk saves no energy by putting it into standby compared to leaving it idle for the entire t_{be} period. However, different prefetching levels (and thus different levels of energy savings) can be achieved by deploying a buffer capacity larger than the break-even buffer. We express this by a scaling parameter called $\alpha \geq 1$ that scales up the primary buffer capacity based on the break-even buffer capacity as the designer chooses based on his budget. We express the capacity of the primary buffer (B_{pm}) as follows:

$$B_{\text{pm}} = \alpha \cdot B_{\text{be}} \tag{C.3}$$

To prevent underrun, we must guarantee $B_{\text{pm}} \geq B_{\text{rt-d}}$, where $B_{\text{rt-d}}$ refers to the real-time buffer capacity of the disk according to Equation (C.1). Hence

$$\alpha \geq \max(1, \frac{B_{\text{rt-d}}}{B_{\text{be}}})$$

C.3.3 The Secondary Buffer Capacity

In HYBBA, we envision a Flash memory located close to the disk drive. That is, it will be interfaced via the I/O subsystem, which is shared across several I/O devices. As a consequence, additional buffering (DRAM) is needed to free the resources of the I/O subsystem, so that shared resources on the I/O bus get their fair share. We call this buffer the secondary buffer. The Flash exhibits orders of magnitude shorter latency than the disk. As a result, the secondary buffer can be small.

The smallest capacity of the secondary buffer is equal to the capacity of the real-time buffer of the Flash, $B_{\text{rt-f}}$. In order to relax the load on the I/O system, we use a scaling parameter, β ($\beta \geq 1$), to tune the capacity of the secondary buffer:

$$B_{\text{sc}} = \beta \cdot B_{\text{rt-f}}. \tag{C.4}$$

The refresh power of DRAM scales largely proportionally to its capacity. Therefore, we tune β such that the DRAM energy consumption is traded off for the I/O bus availability. Recall that, unlike in HYBBA, in DISBA and MEMBA the secondary buffer does not exist, since the primary buffer is located close to the processor.

C.4 Energy Consumption

We calculate the energy consumption of the system for each architecture by aggregating the consumed energy of its constituent components. Since the system periodically refills from the backing store (see Figure 7.2), this section derives the energy consumed by the Disk, the Flash, and the DRAM; that is the energy per refill cycle (T_d). The energy consumption of a MEMS-based storage device is calculated as for the disk after substituting its parameters where applicable. The energy due to communication is small enough to be negligible. In fact, measurements taken on our PDA showed communication energy constitutes a few percentages of the energy consumed by the storage device.

C.4.1 Disk Energy Consumption

The average power dissipated by the disk drive (P_d) is equal to E_d / T_d, the energy consumption during a cycle period T_d, where E_d is:

$$E_d = E_{oh} + P_{RW} \cdot t_{RW} + P_{sb} \cdot t_{sb}, \tag{C.5}$$

with:

$$t_{RW} = \frac{B_{pm}}{r_d - r_s}, \quad t_{sb} = \frac{B_{pm}}{r_s} - t_{oh}, \text{ and } T_d = t_{RW} + t_{sb} + t_{oh}$$

Rewriting Equation (C.5), substituting $P_{oh} \cdot t_{oh} = E_{oh}$ yields:

$$E_d = t_{oh} \cdot (P_{oh} - P_{sb}) + t_{RW} \cdot (P_{RW} - P_{sb}) + T_d \cdot P_{sb} \tag{C.6}$$

$$T_d = \frac{B_{pm}}{r_d - r_s} \cdot \frac{r_d}{r_s} \tag{C.7}$$

with $r_d > r_s$. The disk energy E_d of Equation (C.6) is determined by three factors. The first two terms favor small overhead energy ($t_{oh} \cdot P_{oh}$) and small active energy ($t_{RW} \cdot P_{RW}$). The third term scales the actual standby power by the cycle period. Observe that the T_d scaling of Equation (C.7) is a nonlinear function, see Figure C.4.

C.4.2 Flash Energy Consumption

Flash has two operation states: active and standby. In the active state data are read from and written to the Flash (dissipating $P_{f\text{-}RW}$ [J/s]). In the standby, state the interface awaits requests (dissipating $P_{f\text{-}sb}$ [J/s]). Flash has no startup, idle, and shutdown states, since it is a solid-state memory. The average power dissipated by the Flash (P_f) is computed from the energy per refill cycle (T_d).

$$E_f = E_{f\text{-}RW} + E_{f\text{-}sb}$$

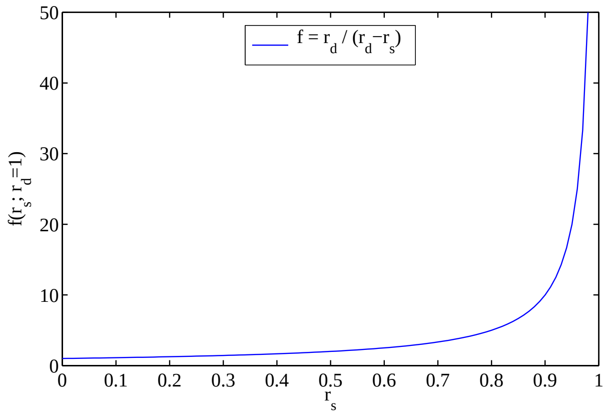


Figure C.4: Nonlinear T_d scaling of the standby power, where $f(r_\mathrm{s}; r_\mathrm{d} = 1) = \frac{r_\mathrm{d}}{r_\mathrm{d}-r_\mathrm{s}}$. As the streaming demand increases, the possibility to go into standby to save energy becomes harder.

$E_\text{f-RW}$ is the energy consumed to write in and then read out an amount of data of size B_pm in $t_\text{f-RW}$ time units. Assuming the Flash has equal read and write bandwidth, r_f, as well as read and write power, $P_\text{f-RW}$, we calculate E_f as follows:

$$E_\mathrm{f} = P_\text{f-RW} \cdot t_\text{f-RW} + P_\text{f-sb} \cdot t_\text{f-sb} \tag{C.8}$$

where

$$t_\text{f-RW} = \frac{B_\mathrm{pm}}{r_\mathrm{d} - r_\mathrm{s}} + \frac{B_\mathrm{pm}}{r_\mathrm{f}}, \quad t_\text{f-sb} = T_\mathrm{d} - t_\text{f-RW}$$

and the throughput constraints are:

$$r_\mathrm{f} > r_\mathrm{d} > r_\mathrm{s}$$

Equation (C.8) is rewritten as follows:

$$E_\mathrm{f} = t_\text{f-RW} \cdot (P_\text{f-RW} - P_\mathrm{sb}) + T_\mathrm{d} \cdot P_\mathrm{sb} \tag{C.9}$$

Here again, small active energy of the total energy consumed by Flash is favorable and the standby energy scales nonlinearly with T_d (Equation (C.7) and Figure C.4).

C.4.3 DRAM Energy Consumption

DRAM consumes energy to retain data and to access (i.e., read/write) data. The retention energy of the DRAM scales proportionally to its capacity. The access energy depends on the access pattern. We refer the reader to a technical report by Micron [84] that details the calculation of the DRAM energy. We implemented the Micron power calculator in our evaluation tool to calculate the energy consumption for different DRAM capacities and access patterns in the three architectures.

This section presented complete models for the three streaming architectures. The next section, extends the models to account for a fraction of best-effort traffic.

C.5 Servicing Best-Effort Requests

Streaming architectures accommodate servicing a small amount of best-effort traffic in addition to the prominent streaming traffic. Best-effort activities include, but are not limited to, loading the Operating System libraries, running application binaries, and executing file system operations. In such predominantly streaming architectures, the disk drive, as a backing store, has to provide access to best-effort data as well as streaming data.

This section extends the analytical models provided in the previous two sections to account for best-effort data access. We express the amount of time the disk drive spends accessing best-effort data t_{bs} as a percentage of the refill period T_{d}, represented by a parameter $\gamma \geq 0$, which yields $t_{\text{bs}} = \gamma \cdot T_{\text{d}}$. Consequently $T'_{\text{d}} = T_{\text{d}} \cdot (1+\gamma) + \epsilon$, where ϵ represents the amount of time needed to buffer more data to account for unavailability of the disk (or the MEMS-based storage device in MEMBA) due to best-effort service. Since streaming is the prominent activity, we assume that γ would be relatively small, say $0 \leq \gamma \leq 0.1$. That said, this has no implication on our analytical study, but just on its orientation toward streaming architectures.

C.5.1 Design Issues

We scale the best-effort service period based on the available primary buffer capacity. The reason is that the longer the disk is in standby due to a larger primary buffer, the more outstanding best-effort requests can be serviced.

In the diagram of Figure C.5, best-effort requests to the disk drive are serviced after streaming requests and before spinning the disk down. The order is irrelevant for the buffer capacities and energy consumption. Best-effort data are routed from the disk drive directly to a best-effort store, typically a DRAM. Thus, best-effort data bypasses the primary buffer (as well as the secondary streaming buffer in HYBBA). Although in practice the best-effort store and the primary/secondary buffer can be realized by just one physical module, we logically separate them in order to model accurately the energy consumption of the streaming primary/secondary buffer.

We do not save substantial energy on best-effort data by caching it in Flash in HYBBA, given its relatively small fraction ($\gamma \leq 0.1$). Hence, best-effort data

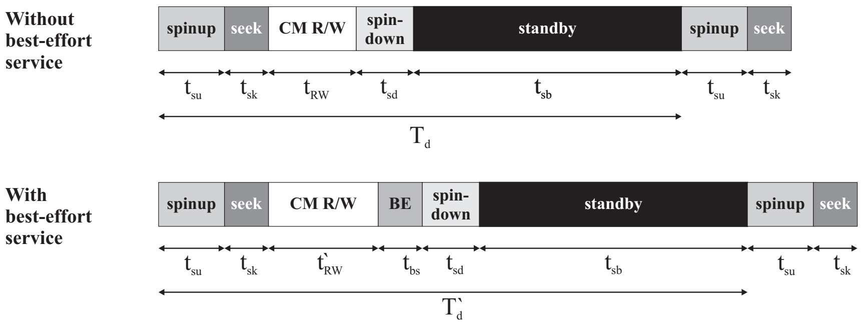


Figure C.5: Extending the pure refill cycle with a best-effort slack to service best-effort data from the backing store. As a consequence, more streaming data should be buffered to keep the same energy-saving level.

are stored elsewhere, consuming a marginal amount of energy. That said, in other mixed-media environments of larger amounts of best-effort data, our buffering technique can be combined with existing best-effort techniques, like Bisson *et al.'s* [87] technique, to reduce the energy consumption for both types of media.

C.5.2 Modeling the Best-Effort Demand

Reserving a service slack of t_{bs} time units in T_{d} for best-effort service shortens the standby time of the disk and thus decreases energy saving. To compensate for the energy loss, we extend T_{d} by enlarging the primary buffer capacity by $t_{\text{bs}} \cdot r_{\text{s}}$, resulting in a standby period length as if no best-effort service exists.

The best-effort period is a percentage of the streaming period ($t_{\text{bs}} = \gamma \cdot T_{\text{d}}$). Therefore, we calculate T_{d} first as shown in Section C.4.1, assuming no best-effort traffic. Then, we calculate the new capacity of the primary buffer and thus the new energy consumption of every component.

We enlarge the primary buffer capacity calculated in Section C.3.2. The buffer is enlarged by $t_{\text{bs}} \cdot r_{\text{s}}$, the amount of data that the buffer has to supply while the disk is servicing best-effort requests. The primary buffer capacity becomes:

$$\begin{aligned} B'_{\text{pm}} &= B_{\text{pm}} + t_{\text{bs}} \cdot r_{\text{s}} \\ &= \alpha \cdot B_{\text{be}} + \gamma \cdot T_{\text{d}} \cdot r_{\text{s}} \end{aligned} \tag{C.10}$$

Here, T_{d} corresponds to the previously calculated refill period, without any best-effort services (see Equation (C.7) of Section C.4.1). From Figure C.5 we observe that only the read/write time changes, whereas the standby time does not change, because only the overhead energy is compensated for as in the best-effort–free case.

The active time period is updated as follows:

$$\begin{aligned} t'_{\mathrm{RW}} &= \frac{B'_{\mathrm{pm}}}{r_{\mathrm{d}} - r_{\mathrm{s}}} \\ &= t_{\mathrm{RW}} + t_{\mathrm{bs}} \cdot \frac{r_{\mathrm{s}}}{r_{\mathrm{d}} - r_{\mathrm{s}}}, \end{aligned}$$

which yields an average disk power dissipation per period T'_{d}, after appropriate substitution of $E'_{\mathrm{d}} = P'_{\mathrm{d}} \cdot T'_{\mathrm{d}}$, as:

$$\begin{aligned} T'_{\mathrm{d}} &= T_{\mathrm{d}} + (t'_{\mathrm{RW}} - t_{\mathrm{RW}}) + t_{\mathrm{bs}} \\ &= T_{\mathrm{d}} + t_{\mathrm{bs}} \cdot \frac{r_{\mathrm{d}}}{r_{\mathrm{d}} - r_{\mathrm{s}}} \qquad \text{(C.11)} \\ E'_{\mathrm{d}} &= E_{\mathrm{oh}} + P_{\mathrm{RW}} \cdot t'_{\mathrm{RW}} + P_{\mathrm{RW}} \cdot t_{\mathrm{bs}} + P_{\mathrm{sb}} \cdot t_{\mathrm{sb}} \\ &= E_{\mathrm{d}} + P_{\mathrm{RW}} \cdot t_{\mathrm{bs}} \cdot \frac{r_{\mathrm{d}}}{r_{\mathrm{d}} - r_{\mathrm{s}}} \qquad \text{(C.12)} \end{aligned}$$

The last term of the disk energy consumption involves a scaled version of the minimum required energy to service best-effort requests: $P_{\mathrm{RW}} \cdot t_{\mathrm{bs}}$. The disk energy to service best-effort requests is included for the sake of a fair comparison, although it is, strictly speaking, not related to the streaming energy.

Titles in the IPA Dissertation Series since 2005

E. Ábrahám. *An Assertional Proof System for Multithreaded Java - Theory and Tool Support-*. Faculty of Mathematics and Natural Sciences, UL. 2005-01

R. Ruimerman. *Modeling and Remodeling in Bone Tissue.* Faculty of Biomedical Engineering, TU/e. 2005-02

C.N. Chong. *Experiments in Rights Control - Expression and Enforcement.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-03

H. Gao. *Design and Verification of Lock-free Parallel Algorithms.* Faculty of Mathematics and Computing Sciences, RUG. 2005-04

H.M.A. van Beek. *Specification and Analysis of Internet Applications.* Faculty of Mathematics and Computer Science, TU/e. 2005-05

M.T. Ionita. *Scenario-Based System Architecting - A Systematic Approach to Developing Future-Proof System Architectures.* Faculty of Mathematics and Computing Sciences, TU/e. 2005-06

G. Lenzini. *Integration of Analysis Techniques in Security and Fault-Tolerance.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-07

I. Kurtev. *Adaptability of Model Transformations.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-08

T. Wolle. *Computational Aspects of Treewidth - Lower Bounds and Network Reliability.* Faculty of Science, UU. 2005-09

O. Tveretina. *Decision Procedures for Equality Logic with Uninterpreted Functions.* Faculty of Mathematics and Computer Science, TU/e. 2005-10

A.M.L. Liekens. *Evolution of Finite Populations in Dynamic Environments.* Faculty of Biomedical Engineering, TU/e. 2005-11

J. Eggermont. *Data Mining using Genetic Programming: Classification and Symbolic Regression.* Faculty of Mathematics and Natural Sciences, UL. 2005-12

B.J. Heeren. *Top Quality Type Error Messages.* Faculty of Science, UU. 2005-13

G.F. Frehse. *Compositional Verification of Hybrid Systems using Simulation Relations.* Faculty of Science, Mathematics and Computer Science, RU. 2005-14

M.R. Mousavi. *Structuring Structural Operational Semantics.* Faculty of Mathematics and Computer Science, TU/e. 2005-15

A. Sokolova. *Coalgebraic Analysis of Probabilistic Systems.* Faculty of Mathematics and Computer Science, TU/e. 2005-16

T. Gelsema. *Effective Models for the Structure of pi-Calculus Processes with Replication.* Faculty of Mathematics and Natural Sciences, UL. 2005-17

P. Zoeteweij. *Composing Constraint Solvers.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-18

J.J. Vinju. *Analysis and Transformation of Source Code by Parsing and Rewriting*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-19

M.Valero Espada. *Modal Abstraction and Replication of Processes with Data*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2005-20

A. Dijkstra. *Stepping through Haskell*. Faculty of Science, UU. 2005-21

Y.W. Law. *Key management and link-layer security of wireless sensor networks: energy-efficient attack and defense*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-22

E. Dolstra. *The Purely Functional Software Deployment Model*. Faculty of Science, UU. 2006-01

R.J. Corin. *Analysis Models for Security Protocols*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-02

P.R.A. Verbaan. *The Computational Complexity of Evolving Systems*. Faculty of Science, UU. 2006-03

K.L. Man and R.R.H. Schiffelers. *Formal Specification and Analysis of Hybrid Systems*. Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2006-04

M. Kyas. *Verifying OCL Specifications of UML Models: Tool Support and Compositionality*. Faculty of Mathematics and Natural Sciences, UL. 2006-05

M. Hendriks. *Model Checking Timed Automata - Techniques and Applications*. Faculty of Science, Mathematics and Computer Science, RU. 2006-06

J. Ketema. *Böhm-Like Trees for Rewriting*. Faculty of Sciences, VUA. 2006-07

C.-B. Breunesse. *On JML: topics in tool-assisted verification of JML programs*. Faculty of Science, Mathematics and Computer Science, RU. 2006-08

B. Markvoort. *Towards Hybrid Molecular Simulations*. Faculty of Biomedical Engineering, TU/e. 2006-09

S.G.R. Nijssen. *Mining Structured Data*. Faculty of Mathematics and Natural Sciences, UL. 2006-10

G. Russello. *Separation and Adaptation of Concerns in a Shared Data Space*. Faculty of Mathematics and Computer Science, TU/e. 2006-11

L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices*. Faculty of Science, Mathematics and Computer Science, RU. 2006-12

B. Badban. *Verification techniques for Extensions of Equality Logic*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2006-13

A.J. Mooij. *Constructive formal methods and protocol standardization*. Faculty of Mathematics and Computer Science, TU/e. 2006-14