國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

快閃儲存系統上之資料配置研究

Analysis of data allocation on multi-bank flash storage system

李苡嬋

Li Yi-Chan

指導教授：楊佳玲 博士

Advisor: Yang Chia-Lin, Ph.D.

中華民國 98 年 7 月

July, 2009

國立臺灣大學碩士學位論文
口試委員會審定書

快閃儲存系統上之資料配置研究

An extensive study of data allocation on multi-bank Flash storage system

本論文係李苡嬋君（學號 R96922122）在國立臺灣大學資訊工程學系完成之碩士學位論文，於民國 98 年 7 月 30 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

[illegible]
（指導教授）

[illegible]　　[illegible]

系 主 任　呂育道

謝詞

首先感謝恩師楊佳玲老師兩年來的悉心指導。老師思慮縝密、治學嚴謹、教學認真，在學生學習的過程中，不斷提供想法上的刺激，引導學生思考問題的本質。除了研究以外，老師也常藉與企業合作的機會，教導學生許多工作上需注意的細節，讓學生除了研究知識之外，對職場上的進退也多了一些了解。日常生活中，楊老師更時常與學生分享生活上的見解與經驗，讓學生獲益良多。在此獻上最誠摯的感謝。此外，也感謝口試委員張立平老師與阮聖彰老師，於口試期間提供的寶貴意見，使得本論文更臻完備。

研究所的學生生活已走進尾聲。感謝所上老師們對課程的精心安排規畫，讓學生於資訊工程領域獲益良多；感謝一直很耐心為我解惑的學長姐，使我能一步一步突破盲點；感謝一路陪伴我的同學與系排學弟妹，在每次遇到挫折時你們總是能幫我充電；感謝我的室友們，讓我雖然隻身在台北念書，仍然有一個溫暖的家。特別感謝身邊的好友們，你們的支持與鼓勵，才使我能堅持到最後。

最後要感謝我親愛的家人們，無論我想做什麼，您們總是無條件地支持我。感謝您們的栽培，讓我能一直無後顧之憂地專注於學業。我會謹記您們的用心，在生活上、工作上、任何一件事情上，都腳踏實地好好的努力，不辜負您們的期望。

學生　李苡嬋　謹識

摘要

快閃記憶體（Flash Memory）已被廣泛使用於嵌入式系統各種可攜帶是電子配備。隨著技術的精進，快閃記憶體的價格不斷下降、容量也不斷增加，因此也逐漸開始應用在各式大型儲存系統上，如預備用於硬碟的固態硬碟(Solid State Disk)。作為儲存設備的元件，效能和壽命是很重要的。在效能方面，因為快閃記憶體沒有任何需要移動的元件，固態硬碟常使用的增進效能方式是將多顆快閃記憶體晶片透過匯流排的連接、讓其可以同時平行運作；此組織方式常稱為多晶片(Multi-chip)或多儲存庫(Multi-bank)快閃儲存系統。在壽命方面，快閃式記憶體的壽命受限於它的清除次數。在多晶片快閃儲存系統上，不同的資料配置方式，會影響快閃記憶體清除次數。

此論文即在分析不同的資料配置方式配合不同的晶片組織架構所造成的清除次數與效能效果。在清除次數方面，由於用戶端(Host)給予的一個要求內的資料之間常有相關性(locality)，因此將屬於同一個要求的資料寫在一起，可以增加清除效能並降低清除次數。而在效能方面，如果於系統不忙碌的狀態下，盡量將資料分散寫在不同晶片上，可以提高效能；但是當系統非常忙碌的情況下，將資料集中寫入和分散寫入的效能非常相近，反而集中寫入減少清除動作，還有可能使效能增加。

我們建立了一個使用模擬追縱導向(Trace-driven)方式的模擬器，模擬快閃儲存系統的運作，藉此觀察不同的資料配置在不同的晶片組織架構下的效果。

關鍵字 – 資料配置策略(data allocation policy)、多儲存庫(multi-bank)、快閃儲存系統(Flash storage system)

Abstract

In the field of embedded systems and consumer electronics, Flash memory has emerged as an excellent storage system. As prices continue to decline and capacities increase, NAND Flash SSD is showing the potential to substitute Hard Disk Drives. As a secondary storage, the reliability and performance issues are important. The lifetime of Flash-based SSD is limited by the erase count of Flash blocks. Under the multi-package Flash organization, different data allocation policies can lead to different number of erase operations. In this thesis, the effect on performance and erase count resulting from different data allocation policies and Flash architectures is studied.

In terms of erase count, preserving data locality by keeping request un-striped leads to less erase count. In terms of performance, the experimental results show that if the workload is light, stripe request to different banks effectively reduces response time. However, when the workload is relatively heavy to the serving Flash architecture, keeping request un-striped still has comparable performance or even better by the assistance of erase count reduction.

Keywords — data allocation policy, multi-bank, Flash storage system

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Flash has several attractive features compared to hard disks, such as faster read access, low power consumption, small size, shock resistance and reliability compared to hard disks. Therefore, Flash-based solid state disks are rapidly becoming a popular alternative to hard disk drives as permanent secondary storage[26]. Today, Flash-based SSD is a mature technology for mainstream personal mobile computing, and is appealed to server and supercomputer market.

However, Flash also has some properties that has prevented a complete replacement of hard disks. These characteristics including the lower bit density compared to hard disk drives, a limited number of erase operation, and the comparably slow write performance. On each of these fronts, Some efforts have been made. For low bit density, the Multi-level cell(MLC) technology doubles the bit density and lower the bit cost[23]. Since 2000, the capacity of NAND Flash memory has been doubled every year[21], and is expected to continue at that rate until 2010. Multi-package architecture in SSD has developed to overcome the slow write speed. By properly coordinate the memory components and operating in parallel, Flash can achieve higher performance than the bandwidth and operation rate of a single Flash chip[1, 2]. In addition, technological enhancements have improved Flash endurance. For the issue of prolong Flash's lifetime, high-end SSD adopts a write buffer or disk cache to reduce write operations. In Flash translation layer(FTL) design, many wear-leveling algorithms were proposed to balance erase counts of SSD blocks.

SSD seems excellent from the above depiction. However, the design of hardware architecture and the details of the FTL are not well known, and these are closely related to the actual SSD performance. In this thesis, the evaluation focuses on studying the effectiveness of the combination of many existing methodologies. How does different FTL design affects the performance and lifetime of a SSD? Does different Flash organization alter the outcomes? I found the following:

1. Under slight workload, serving one request in parallel greatly reduces the response time. However, when the workload becomes heavier, the benefit from spreading one request to many banks becomes less significant.

2. Keeping pages of one request together preserves the request's locality which results in better garbage collection efficiency. The erase operation can be reduced in this way, and lifetime of SSD can be prolonged.

3. If we have limited banks, the benefit of parallel execution reduced. In this case, reducing the parallelism in exchange for less erase count is a good choice. If the workload is extremely heavy, performance can even be improved by

reducing erase operations.

To have a better understand to the interplay between the hardware and software design, I constructed a cycle-accurate, trace-driven simulator. The interconnection and the Flash banks are modeled, and basic functionalities of FTL are implemented. In this evaluation, different data allocation approaches and Flash organizations are investigated, and their influence to response time and number of erase operations is discussed.

The rest of the thesis is organized as follows. Chapter 2 introduces the background knowledge of Flash memory and SSD basics. Chapter 3 discusses the tradeoffs of the target data allocation policies. Chapter 4 provides the experimental methodology and results. Related works are in Chapter 5, and Chapter 6 concludes this thesis.

Chapter 2

Background

2.1 Flash Characteristics

Flash memory is a non-volatile computer memory that can be electrically erased and reprogrammed. There are two types of Flash memories, NOR and NAND [20]. NOR supports random accesses in bytes, and is mainly used for storing code. NAND, which has a different internal structure, is designed for data storage with denser capacity, and only allows access in units of sectors. The comparison between NAND Flash memory and NOR Flash memory is presented in Table 2.1. We focus this research on NAND Flash memory because most SSDs available on the market are based on NAND Flash memories. NAND Flash memory can be classified into two categories, Single-Level Cell(SLC) and Multi-Level Cell(MLC) NAND. A SLC Flash memory cell stores only one bit, while a MLC Flash memory cell can store two more more bits. SLC has lower operational latency and higher endurance, while the capacity of MLC doubles that of SLC.

Figure 1 shows a typical architecture of a NAND Flash memory [7]. A

Factor	NOR Flash	NANS Flash
Read Performance	Very Fast	Moderate
Write Performance	Slow	Fast
Erase Performance	Very Slow	Fast
Random access	Possible	Impossible
eXecute In Place (XIP)	Supported	Not Supported
Capacity	Small	Large
Cost ($/MB)	High	Low

Table 2.1: Comparison of NOR Flash and NOR Flash. [16, 5]

typical NAND Flash memory consists of multiple blocks, and a block is partitioned into a fixed number of pages. The typical block size and page size are 256KB and 4KB [7].

There are three operations in Flash memory: read, write, and erase. The read and write operation are done in a unit of page, while the unit of erase is block. Since a write operation can only change bits from 1 to 0, in-place update of data is not allowed. To avoid costly erase on every write, Flash memory performs out-place updates. That is, data is written to a free page, and the old page is invalidated. Pages with valid data are called live pages or valid pages, while the ones who were invalidated are named dead pages or invalid pages.

After a certain number of writes, free space on Flash memory would be low. Therefore, Flash memory reclaims free page by a process called garbage collection. A garbage collection process first selects a victim block, copies all live pages on the block to free spaces, and then erases the victim block. Live page copying is considered as the overheads of garbage collection process. Therefore, a major consideration of victim block selection is to minimize these overheads.

The number of erase operations that can be done on a block is limited (i.e., 100K for blocks on a SLC Flash chip [7] and 10K on a MLC Flash chip [1]). If the number of erase operation on a block exceeds the threshold, the worn-out block could suffer from frequently write errors. Therefore, the garbage collection process should be avoided not only for better performance, but also longer Flash memory lifetime.

2.2 NAND Flash Package Organization and Commands

The main Flash chip manufacturers now are Samsung and Micron. The organization of Flash packages between two manufactory are slightly different, which are summarized as the following subsection. In this thesis, the abstraction of a bank is used to indicate the unit that can do operations independently for general evaluation.

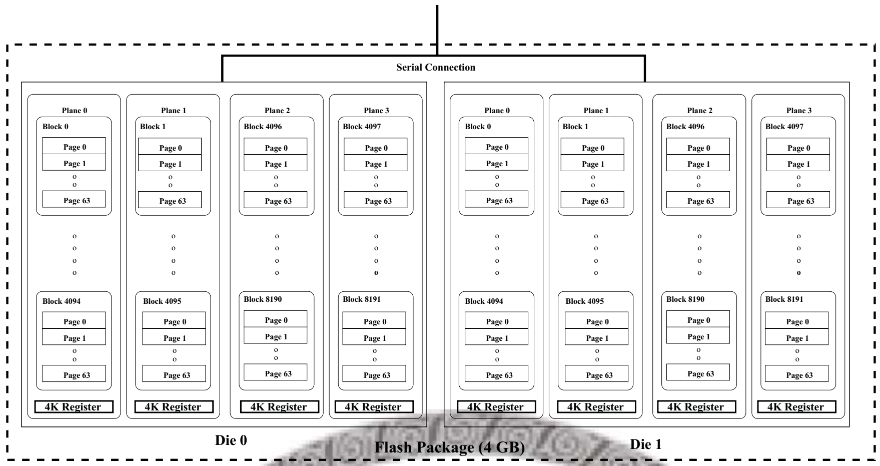


Figure 2.1: Samsung 4GB Flash Package.

2.2.1 Samsung Flash Package

Figure 2.1 shows the schematic of a Samsung Flash package[1]. A NAND Flash package consists of dies (also called chips). Dies in a Flash package shares an 8-bit serial I/O bus and a number of common control signals. The chip enable and ready/busy signals can be shared or separated, depending on the cost consideration. If two dies have separate chip enable and ready/busy signals, they can operate independently. That is, one can accept commands and data while the other is carrying out another operation. The package supports interleaved commands to manage these independent operations between dies.

Each die within a package contains several planes. A plane consists of blocks with fix number of pages, and a data register with size of one page. The functional diagram for a Flash die is shown in Figure 2.2[7]. To issue a command, the command if first written into the command register. Then, according to the row address and column address given by controller, the

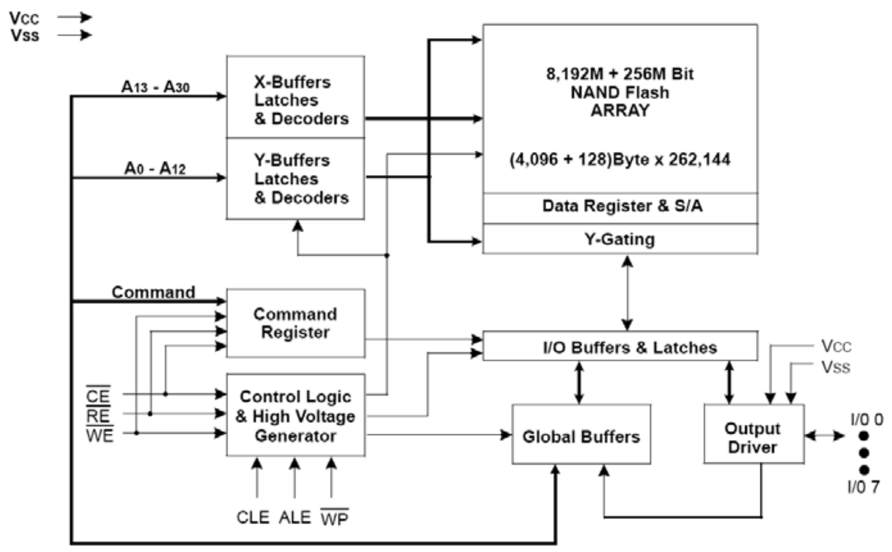

Vcc
Vss
A13 - A30
A0 - A12
Command
X-Buffers Latches & Decoders
Y-Buffers Latches & Decoders
Command Register
Control Logic & High Voltage Generator
8,192M + 256M Bit NAND Flash ARRAY
(4,096 + 128)Byte x 262,144
Data Register & S/A
Y-Gating
I/O Buffers & Latches
Global Buffers
Output Driver
CE
RE
WE
CLE ALE WP
I/O 0
I/O 7

Figure 2.2: Samsung Flash Chip Functional Diagram.

target page or block is decoded by X/Y-Buffers Latches and Decoders. If the command is a read command, the complete page data will then be loaded to data register in 25μs, and passed out through the serial interface to controller buffer in 100μs. If the command is a write command, the written data will first be written to the data register in 100μs, and then be programmed into the page in 200μs. If the command is a block erase command, the target block will be erased in 1.5ms. Samsung also implements a copy-back command. In a copy-back command, the data is read to the data register and program back to the destination page without crossing the serial pins.

In addition to conventional single plane operations(i.e., page read, page program, block erase, ... etc.), Samsung Flash chips also support two-plane commands. The two-plane commands can be executed on either plane-pairs 0&1 or 2&3, but not across other combinations(such as plane 0&2).

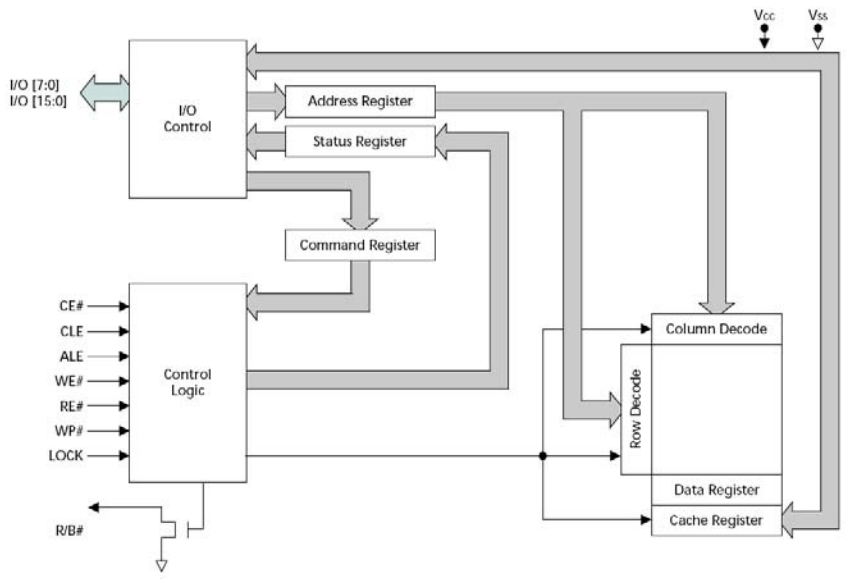


Figure 2.3: Micron Flash Chip Functional Diagram.

2.2.2 Micron Flash Device

In Micron Flash chip(named 'device' by Micron), a 'cache mode' concept is proposed. A Micron device contains blocks with fix number of pages, data register, and a cache register as shown in Figure 2.3. In the cache programming mode, data is first copied into the cache register, and then into the data register in $3\mu s$. Programming starts when the data is loaded into the data register. At the same time, the cache register becomes available for loading additional data. Other operational latencies are similar to Samsung Flash chip.

2.3 SSD Basics

NAND-based SSDs are constructed from an array of Flash packages. A generalized block diagram is depicted in Figure 2. A SSD contains host interface logic, SSD controller, and an array of Flash packages. The host interface logic support physical host interface connection(for example, USB, PCI Express, SATA)[22, 13, 12]. The SSD controller consists of a processor, buffer-manager, and multiplexer. The processor manages the request flow and takes care all policies required in FTL layer, such as logical block map, garbage collection policy, and wear leveling policy. An internal buffer manager holds pending and satisfied requests along the primary data path. A multiplexer implements the MTD layer, which emits commands and handles transport of data along the serial connections to the Flash packages.

2.3.1 Logical Block Map and Data Allocation Policy

Since writes to Flash cannot be performed in place as on a rotating disk, SSD must maintain some form of mapping between logical-disk block address(LBA) and physical Flash location. The logical block map is usually held in volatile memory(for example, SRAM) when SSD is executing, and reconstructed from stable storage at startup time. The mapping granularity decides the volatile memory requirement. Larger mapping granularity has little SRAM requirement, but has to deal with read-modify-write problem when the written size is smaller than the mapping granularity. In this thesis, we assume page granularity in address mapping.

The physical Flash location for each write request is decided by the data allocation policy implemented in the SSD controller. For environment such as multi-package SSDs that multiple job can be executed at the same time, data allocation policy plays an important role on driving all components. For example, static striping [4, 24] can spread a long sequential requests to many packages, but may leave some packages idle if the request is not long enough to occupied all packages, even if there is some other requests waiting to be served. On the other hand, dynamic striping dispatches a write command to any free bank, can better utilize all available resources. More elaborate discussions of data allocation policies are left to chapter 3.

2.3.2 Garbage Collection

As mentioned before, garbage collection has two decision to make: victim block selection and destination to copy all live pages. Garbage collection process is triggered when the number of free pages left is less than a specific threshold(also known as cleaning threshold). In the multi-bank Flash organization, to ensure that number of available banks would not decrease, the number of free pages is usually monitored for each bank. In the live page copying process, the location of the source and destination blocks determine the copy method. If the source and destination are on the same plane, the live page copying process can be carried out by copy-back feature without having to transfer them across the serial pins. However, garbage collection process can be accelerated by selecting multiple destinations on different banks, and copy live pages in parallel by concurrently write to the selected banks.

2.3.3 Wear Leveling

Only a limited number of erase operations can be done on a given block, so wear leveling attempts to prolong the life of a Flash device by re-arranging data so that erasures and re-writes are distributed evenly across the blocks. Wear leveling algorithms are commonly triggered during garbage collection or write operations[11, 6, 3]. A conventional wear-leveling mechanism is to migrate cold data to an aged block, which has larger erase count. Since the migration involves several page copies and an erase similar to garbage collection, it can degrade the system performance.

On multi-bank Flash architecture, the wear leveling cluster[24], is also a design issue. Only lifetime of blocks within a wear leveling cluster can be balanced. Therefore, larger wear leveling cluster leads to more evenly distributed lifetime in SSD. However, the migrating overhead will be larger, inducing greater system performance degradation

Chapter 3

Data Allocation Policies on Multi-Bank

There are three operations supported by Flash memory: read, write, and erase. Except for write operations, reads and erases do not need further assignment, for they are already stored in specific locations. For write operations, since the write unit of Flash is a page, every write request that comes to Flash has to be transformed to many page writes. The function of a data allocation policy is to decide the physical Flash location for each page write. Different data allocation policy leads to different load distribution, parallelism, and garbage collection efficiency. Generally, there are two approaches to allocate page writes on multi-bank Flash architecture. The detail of these two approach and their tradeoffs are described as follows.

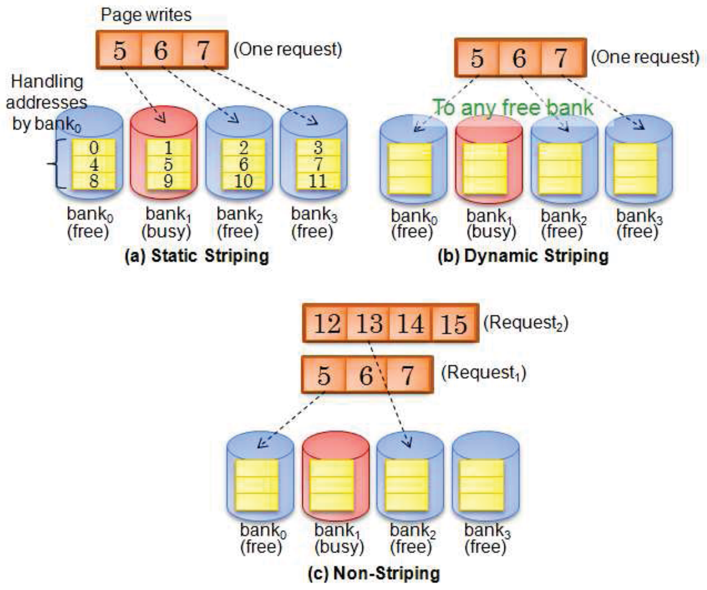


Figure 3.1: Data Allocation Policies on Multi-bank Flash Architecture.

3.1 Striping

The striping approach is to spread page writes of one request to many different banks. By striping, one request can be served by many banks concurrently, thus shortening the response time.

There are two striping mechanisms: static striping and dynamic striping, as discussed in [4, 24]. In static striping, page writes are allocated by their logical address. For example, if we have a total of x banks, bank assignment for each page write equals to LBA(mod x). Shin et al. have studied the performance of various mappings on static striping [24]. In addition to static striping, dynamic striping get rid of the address limitation. In dynamic striping, page write is allocated to any free bank. In this way, banks can be better utilized. The two striping policies are illustrated in Figure 3.1(a),(b). Since dynamic striping have better performance, for the rest of this thesis, the term "striping" refers to dynamic striping.

3.2 Non-Striping

Another approach which works very well with the support of parallel requests is non-striping, which keeps all page writes of one request to one bank, and assign page writes of the following request to another bank. An example of non-striping is illustrated in Figure 3.1(c). If we have two requests to serve, all page writes of the first write request are assigned to one free banks, and the page writes of the second write request are assigned to another free bank if there is any. If the workload is heavy, banks can still be fully utilized in this way.

3.3 Tradeoffs between Striping and Non-Striping

The tradeoffs between striping and non-striping lie in the parallelism and the garbage collection efficiency. Consider the example shown in Figure 3.2 and Figure 3.3 which schemes the simplified outcome(only consider the pure write time) of two requests dispatched by these two policies. Assume two requests arrive simultaneously. With striping, which is illustrated in Figure 3.2(a) the page writes of two requests are scattered to two banks. By spreading the page writes, the average response time for the two requests is 700μs. With non-striping, which is shown in Figure 3.2(b), the average response time is 800μs. In terms of the physical pages arrangement in Flash blocks, non-striping gathers page writes of one request together(i.e., in the same block for most cases). This increase the possibility that pages within a single block

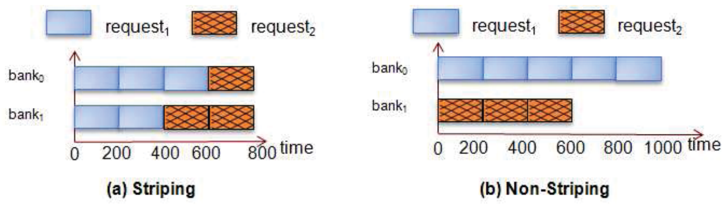


Figure 3.2: The Time Diagram of (a)Striping and (b)Non-Striping Data Allocation Policy.

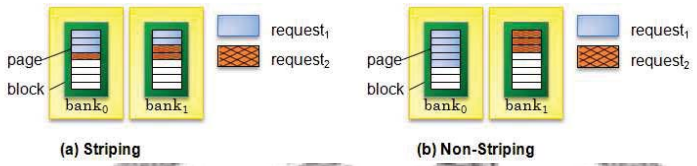


Figure 3.3: The Arrangement in Flash Blocks by (a)Striping and (b)Non-Striping.

be invalidated at similar time. For example, if the host issue request 1 again, the corresponding 5 pages in block 0 will be invalidated at the same time when using non-striping allocation policy, while 3 pages in block 0 and 2 pages in block 1 are invalidated when striping is adopted.

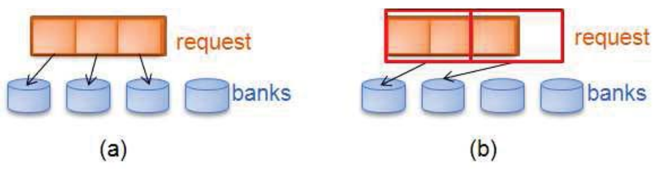


Figure 3.4: Illustration of Different Striping Unit Size. (a) Striping Unit Size = 1 page. (b) Striping Unit Size = 2 pages.

3.4 Striping Unit Size

To examine the trade-off between parallelism and garbage collection efficiency in more detail, a concept ”striping unit size” is introduced. The striping unit size restricts the number of pages that must be allocated together. Figure 3.4 shows an example of striping unit size equals to (a)one page and (b)two pages. In the case of two pages striping unit size, every two page writes are clustered and allocated together. With larger striping unit size, the degree of parallelism decreases, but the garbage collection efficiency may be better.

Chapter 4

Implementation and Evaluation

4.1 Simulation Environment

In order to study the performance and endurance implications of various data allocation policies and Flash organizations, a trace-driven, cycle accurate simulator, as shown in Figure 4.1, is built. Traces are gathered by disk I/O monitor tools(DiskMon in Windows and blktrace in Linux). The trace are recorded by a 4-tuple (command, LBA, length, issue_time). There are two

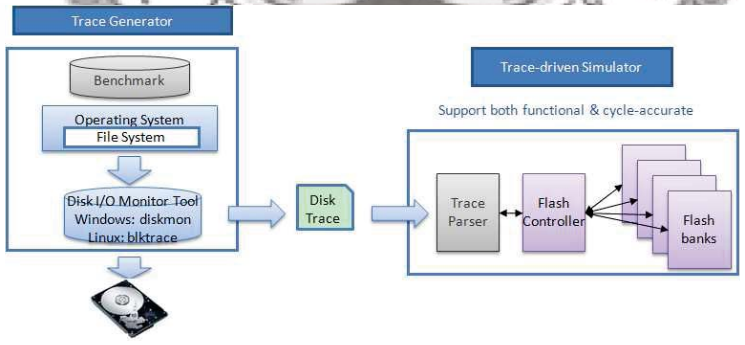


Figure 4.1: Trace Driven Flash Simulator.

Software Implementation

Data Allocation Policies	Dynamic Striping (with various striping unit size Non-Striping
Address Mapping Mechanism	Page-level Mapping
Garbage Collection Policy	Greedy Using Copy-back command
Garbage Collection Threshold	5%
Over-Provision	15%

Default Hardware Configurations(Configurable)

Channel Bandwidth	40MB/s
Number of Banks	4
Number of Channel	4
Number of Banks Per Channel	1
Page Size	4KB
Block size	256KB (64 pages)

Timing Parameters(Configurable)

Page Program Time	200 μs
Page Read Time	25 μs
Block Erase Time	1.5 ms

Table 4.1: Detail Implementation of Flash Simulator.

type of commands: Read and Write. The LBA and length are all in the unit of disk sector(512 bytes). The recorded issue_time is in μs.

The software implementation in Table 4.1 summarized the software implementation of this simulator. Different data allocation policies have been implemented, such as dynamic striping with various striping unit size, non-striping algorithm. In addition to data allocation policies, the greedy policy is chosen for garbage collection policy for it is the most commonly used cleaning policy. In hardware configuration, a multi-bank Flash architecture is implemented. The default configuration is listed in Table 4.1, and are all configurable. The operational latency is followed by[7], and is also summarized in Table 4.1.

Benchmark	SPECWeb	Sysmark	PCMark
I/O Requests Per Second	11.9	27.0	91.0
Avg Request Size (KB)	9.8	24	38.6
Write Request %	89.8	55.2	43.5
Write Data %	87.7	42.1	57.4

Table 4.2: Workload Characteristics.

4.2 Workload Analysis

The objective of this thesis is to evaluate the performance and endurance implication under different data allocation policies and Flash multi-bank organizations. Since most Flash-based storage in mobile platforms(such as MP3 players, digital cameras, PDAs, ...etc) are usually composed of only 1 Flash chip, only server and PC workloads are considered in this thesis. The evaluated traces are selected from three public benchmarks: SPECWeb2005, Sysmark2007, and PCMark2005. The characteristics of selected traces are listed in Table 4.2. Two special properties can be observed from the summary: 1. Major portion(90%) of requests in SPECWeb Banking are write, while for the two other PC traces, around 50%. 2. PCMark has a extremely high I/O request arrival rate. Some detail elaborations and request addresses, length distributions are described as follow.

4.2.1 SPECWeb2005 - Banking

SPECWeb2005 mimics the access patterns of real-world web servers. The trace selected is "Banking", which imitates the trades of online banking. 4 banking transactions are modeled: Account information access or Account

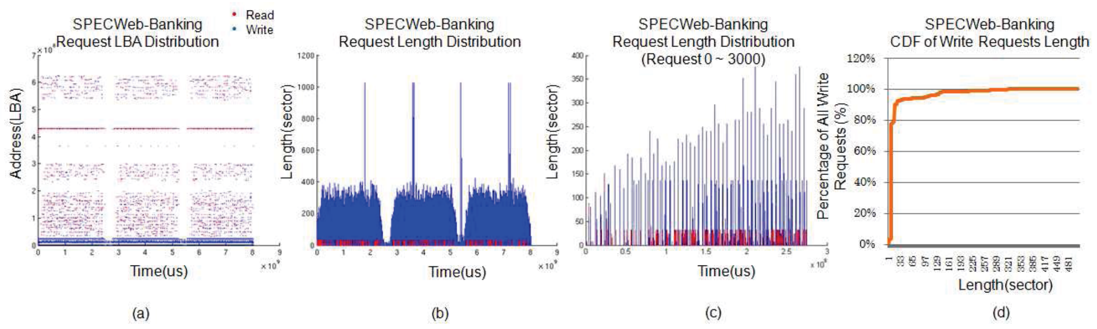


Figure 4.2: Properties of SPECWeb2005 - Banking. (a) The Address Graph. (b) The Length Distribution. (c) The Length Distribution of Request 0 to 3000. (d) The Cumulative Distribution of Write Requests.

Maintenance related, Bill Pay related, Money Transfer, Loan related. Detail transactions and the corresponding returned pages, such as login and welcome message returns few 4KB pages, account summary returns 17KB pages, bill pay returns 15KB pages, ...etc, can be referred from website of SPECWeb2005 [8].

The I/O properties of Banking are depicted in Figure 4.2. Compared to PC traces which will be elaborated next, Banking consists of many short requests. A snapshot for request 0 to request 3000 is taken as shown in Figure 4.2(c). As statistic in Figure 4.2(d), 77% requests are equal or smaller than 4KB, and 90% requests are equal or smaller than 12KB. Most of the rest 10% requests scatter between 12KB and 128KB. The maximum request length is 512KB.

4.2.2 Sysmark2007 - E-learning

Sysmark2007 mimics the access patterns of business users. The trace selected is "E-learning", which imitates the tasks executed in the development of an

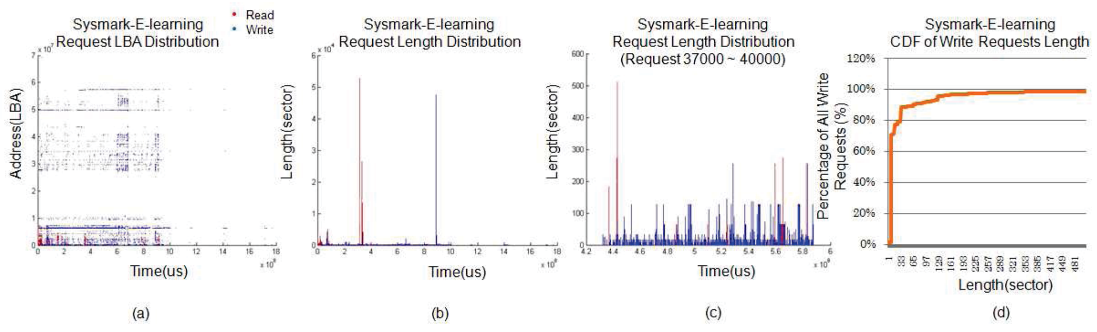


Figure 4.3: Properties of Sysmark2007 - E-learning. (a) The Address Graph. (b) The Length Distribution. (c) The Length Distribution of Request 37000 to 40000. (d) The Cumulative Distribution of Write Requests.

on-line teaching facility. The E-learning scenario integrates rich image, video, and audio content in a web page for presentation of learning materials using programs predominantly from Adobe cumulating in the creation of an Adobe Flash file. Some usage examples are: Microsoft Power Point 2003, Adobe Photoshop, Adobe Flash, Adobe Illustrator.

The I/O properties of E-learning are depicted in Figure 4.3. The footprint of E-learning, as shown in Figure 4.3(a), is more concentrate compared with footprint of Banking. Beside, the workload is bursty, similar to the PC traces gathered in [9]. As mentioned to command type, E-learning has more read requests compared to Banking. These read requests are mingled with write requests, and the overlaps of read and write addresses are significant. When speaking of request length, in addition to some extremely long requests as shown in Figure 4.3(b), most requests are short, like Figure 4.3(c). As statistic in Figure 4.3(d), 70% requests are 4KB, and 10% requests are 16KB.

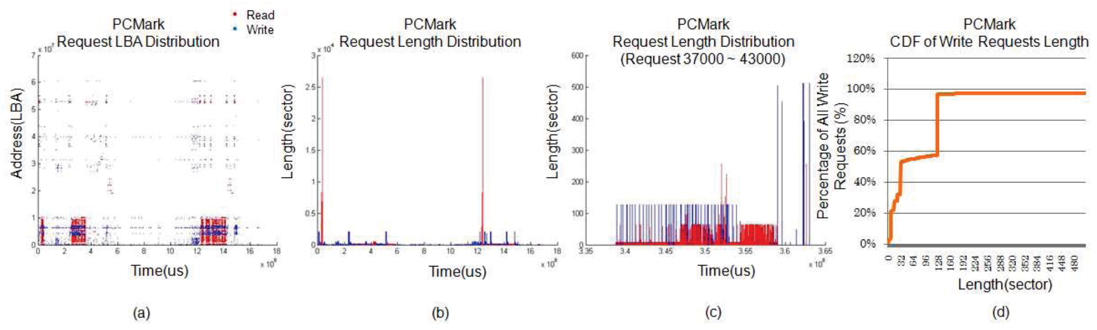


Figure 4.4: Properties of PCMark. (a) The Address Graph. (b) The Length Distribution. (c) The Length Distribution of Request 37000 to 43000. (d) The Cumulative Distribution of Write Requests.

4.2.3 PCMark2005

PCMark2005 developed by the model of home PC usage. The tasks including file(including ordinary files, image, and audio) compression, decompression, encryption, decryption, text editing, web page rendering, video encoding, ...etc.

The I/O properties are depicted in Figure 4.4. Similar to E-learning, the footprint is concentrate. One of the unique properties observed from PCMark is its great overlap of read and write requests as shown in Figure 4.4(a). Moreover, I/O requests are bursty and localized. The other unique property is that PCMark consisted of lots of long requests. Figure 4.4(c) shows the length distribution of request 37000 to 43000. Most read requests are 4KB(8 sectors) and 32KB(64 sectors), and lots of 64KB(128 sectors) write requests. The cumulated distribution of write request length are shown in Figure 4.4(d). About 20% of write requests are 16KB(32 sectors) and 40% of that are 64KB(128 sectors).

4.3 Striping vs. Non-Striping

The tradeoffs lie in the parallelism and the garbage collection efficiency. In this section, the performance impact is evaluated by 2 set of experiments. The first experiment evaluates the performance impact of parallelism. The evaluation shows that if the workload is heavy, the impact of parallelism to response time is mitigated. In this case, non-striping and striping will have similar response time. The second experiment considers both parallelism and garbage collection overhead. The experimental results show that non-striping effectively slower the increase of erase count when multiplying number of banks. On the other hand, when the workload is heavy, non-striping with slighter garbage collection overhead leads to less response time compare to striping.

4.3.1 Concurrency

In this experiment, the garbage collection overhead is set to zero. The assumption behind this experiment is that no garbage collection process is triggered. Therefore, only the concurrency affects system performance. The normalized response time shown in Figure 4.5 are the average response time of non-striping relative to the average response time of striping [1]. If free banks are sufficient for serving requests, the striping mechanism can distribute every page write command to a free bank anytime. In this case, the normalized write response time will be related to average request length, as shown in Figure 4.5(f). However, when banks are not sufficient for serving requests, banks in non-striping are also fully utilized, the normalized write

[1] $\frac{ResponseTime_{Non-Striping}}{ResponseTime_{Striping}}$

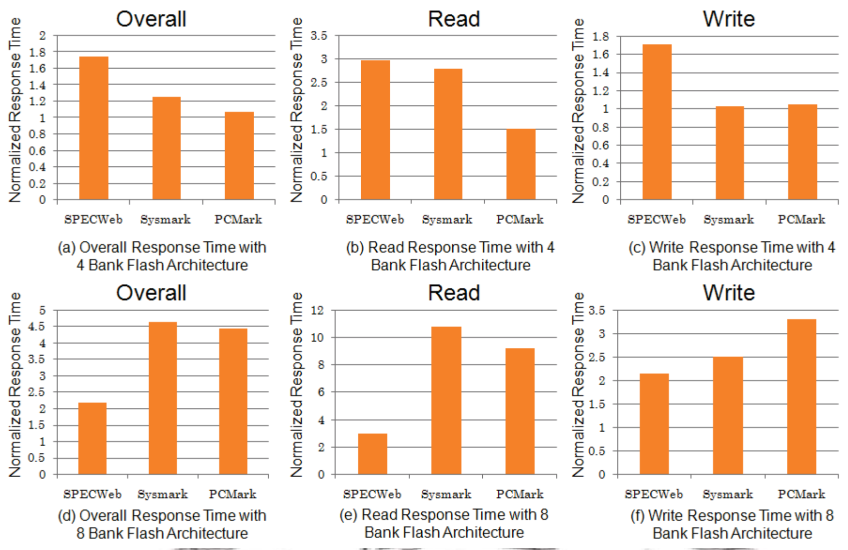


Figure 4.5: Response Time of Non-Striping Mechanism Normalized to Striping without Considering GC Overhead. (a)(b)(c) Are in 4 Bank Flash Architecture. (d)(e)(f) Are in 8 Bank Flash Architecture.

response time will be close to 1 as shown in Figure 4.5(c).

The response time under different banks are summarized in Figure 4.6. The normalized response time shown in Figure 4.6 is the average response time relative to 1 bank. For three selected traces, increasing number of banks leads to significant performance improvement in striping. On the other hand, the performance improvement to non-striping depends on workload. As shown in Figure 4.6(a), a not so heavy workload such as SPECWeb, 2 banks is sufficient for non-striping, further increasing number of banks improve little performance. If workload is heavy such as PCMark, the performance improvement is more significant when there are more available banks, as shown in Figure 4.6(c).

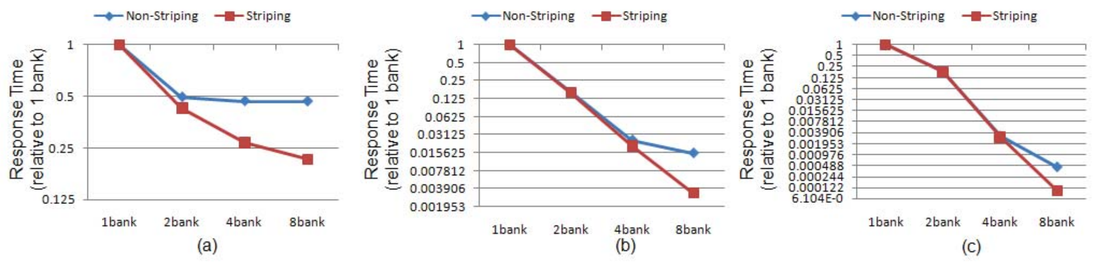


Figure 4.6: Response Time for Different Number of Banks without Considering GC Overhead. (a) SPECWeb2005 - Banking. (b) Sysmark2007 - E-learning. (c) PCMark2005.

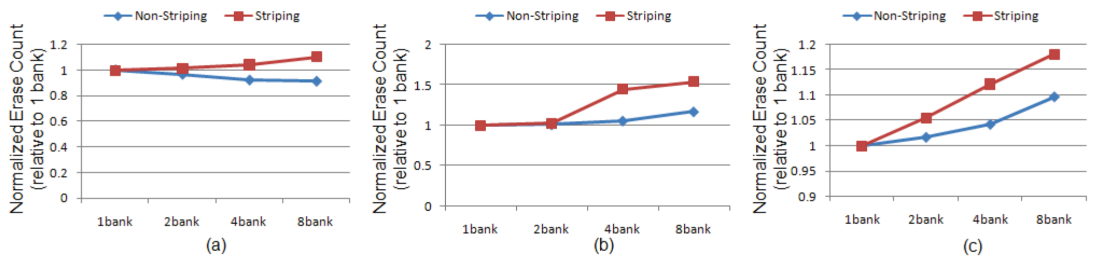


Figure 4.7: Erase Count for Different Number of Banks. (a) SPECWeb2005 - Banking. (b) Sysmark2007 - E-learning. (c) PCMark2005.

4.3.2 Erase Count

Figure 4.7 shows the erase count under different banks. The normalized erase count in Figure 4.7 is relative to erase count of 1 bank. As expected, the growth of erase count by striping is faster than non-striping when increasing number of banks. In addition to this obviously result, one can find that in Sysmark with striping, a large erase count gap happens when number of bank increased from 2 to 4.

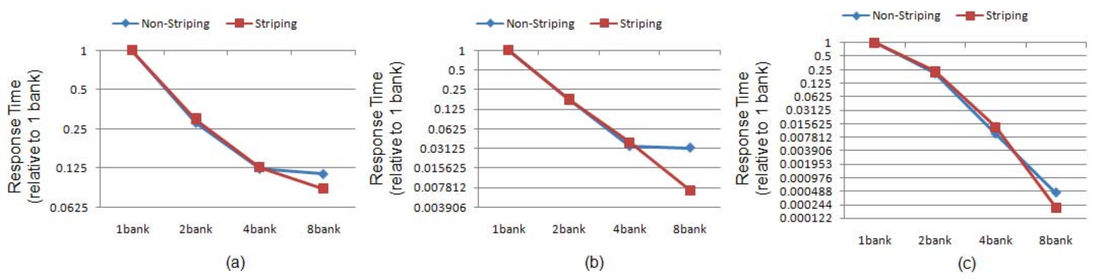


Figure 4.8: Normalized Response Time for Different Number of Banks Considering Garbage Collection. (a) SPECWeb2005 - Banking. (b) Sysmark2007 - E-learning. (c) PCMark2005.

4.3.3 Performance Considering Garbage Collection

The response time of each benchmark under different number of banks is shown in Figure 4.8. If banks are sufficient for serving(for example, 8 bank in the experiments), striping still achieve better performance. However, when number of bank is limited(2 and 4 banks), non-striping outperforms striping.

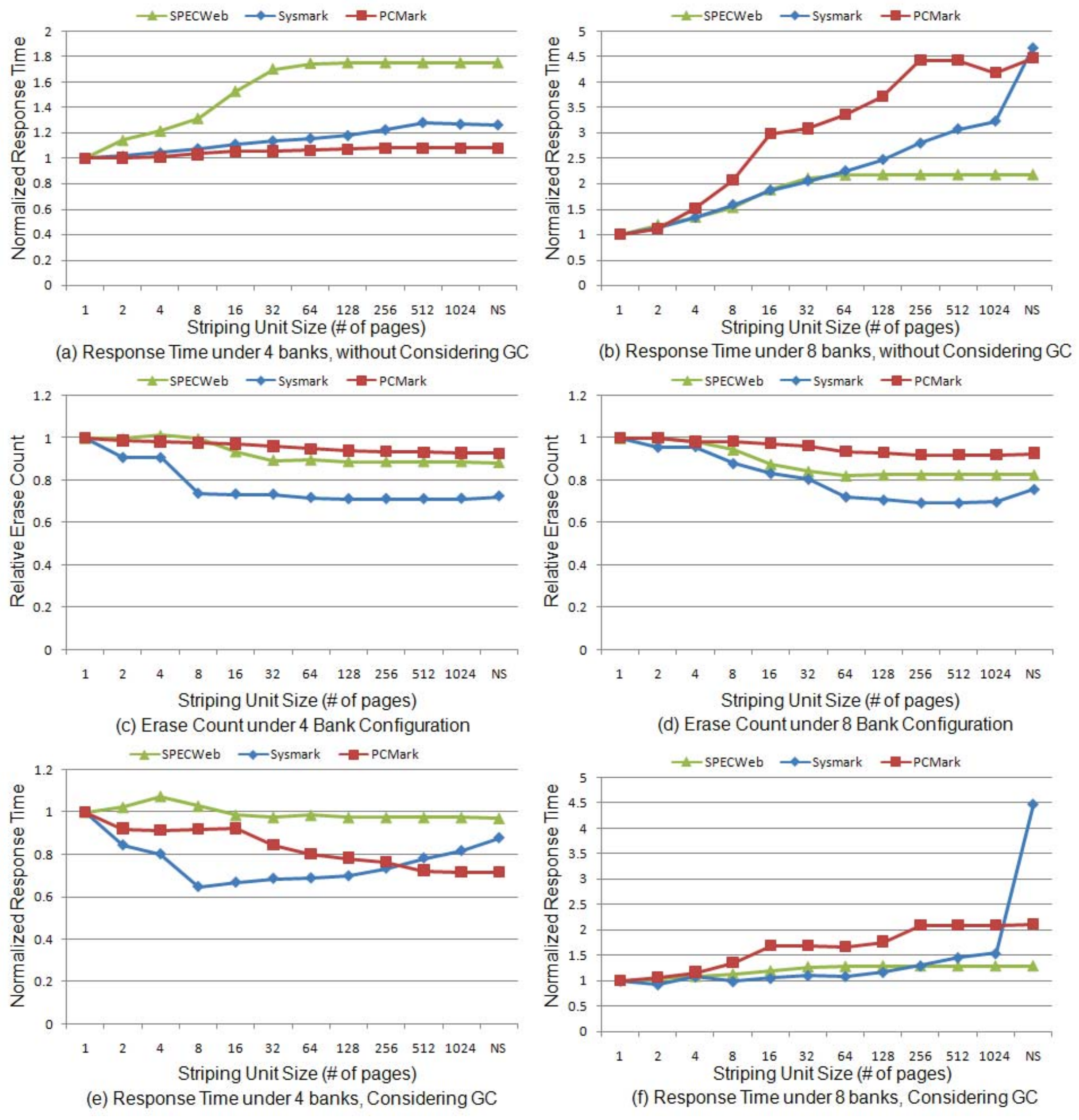
4.4 Different Striping Unit Size

Now we know that striping all page writes to different banks has better parallelism, while keeping all page writes together reduces number of garbage collection. To have a more detail evaluation on parallelism and garbage collection performance, finer striping granularity is examined. Following the flow of Section 4.3, the performance without garbage collection is first exhibited, and then the erase count and performance impact is discussed.

4.4.1 Concurrency

Figure 4.9(a),(b) shows the response time with different striping unit size without considering garbage collection. The response time is normalized to striping unit size = 1. The last point which labeled as NS represents for the result of non-striping.

As expected, larger striping unit size leads to larger response time, but traces with short requests(such as SPECWeb) saturate fast. On the other hand, one can find that Sysmark and PCMark have different curves under 4 bank and 8 bank architecture: little performance degradation under 4 bank but large under 8 bank. The reason is that 4 banks is not sufficient for serving these 2 workload easily. Even with non-striping mechanism, banks are still fully utilized. Therefore, larger striping unit size only results in little performance degradation.



(a) Response Time under 4 banks, without Considering GC

(b) Response Time under 8 banks, without Considering GC

(c) Erase Count under 4 Bank Configuration

(d) Erase Count under 8 Bank Configuration

(e) Response Time under 4 banks, Considering GC

(f) Response Time under 8 banks, Considering GC

Figure 4.9: Experimental Results of Different Striping Unit Size.

4.4.2 Erase Count

Larger striping unit size improves garbage collection efficiency, thus reduces erase count, as shown in Figure 4.9(c),(d). When number of banks increase, striping with small striping unit size generates more garbage collection. Under this circumstance, enlarge striping unit size reduces more erase count, as in the 8 banks architecture.

Note that in both 4 bank and 8 bank Flash architecture, the erase count curve raise from striping unit size equals to 1024 pages to non-striping. The reason is that the write request length in Sysmark varies a lot. The largest write request writing 47480 sectors, or 23MB. Keeping page writes of long request on a particular bank leads to imbalance bank utilization. Banks with higher utilization result in more garbage collection, since the garbage collection process is triggered more frequently.

4.4.3 Performance Considering Garbage Collection

Figure 4.9(e),(f) shows the response time of different striping unit size in 4 bank and 8 bank Flash architecture. The response time is normalize to striping unit size = 1.

For SPECWeb, the erase count reduction appears within striping unit size = 8 to 32. Therefore, the response time follows the trends of response time not considering garbage collection(as in Figure 4.9(a),(b)) when striping unit size is smaller than 8 or larger than 32. For striping unit size between 8 and 32, the reduced erase count helps in suppressing the raise of response time. As in Figure 4.9(f), though the results hints that SPECWeb under this configuration still prefer parallelism than erase count reduction for better performance, the reduced erase count still lessen the performance degradation of larger striping unit size.

For Sysmark with 4 banks, the erase count reduction occurs within striping unit size = 1 to 8. Since this traces is relatively heavy for 4 banks, the response time degradation by less parallelism is not serious as shown in Figure 4.9(a), erase count reduction leads to significant performance improvement as depicted in Figure 4.9(e) withing striping unit size = 1 to 8. However, when the erase count stops decreasing, the response time turns its head back to degrade. Under 8 bank configuration, since the workload is relatively light, reduced erase count only helps in lighten the performance degradation. Moreover, when striping unit size increase from 1024 pages to unlimit(non-striping), the increased erase count deteriorate the performance degradation.

For PCMark, the erase count keeps reducing when enlarge striping unit size. The performance impact is similar to Sysmark: Under 4 banks, the workload is relatively heavy, thus larger striping unit size outperforms small striping unit size by the assistance of erase count reduction; under 8 banks, the performance degradation is alleviated when erase count reduced.

Chapter 5

Related Works

5.1 related work

In the past years, the researchers focus on improving performance by carefully designing each component in FTL. For address-mapping technique, the tradeoffs between SRAM requirement and performance is a widely discussed topic. Block-level mapping was the first solution to reduce SRAM requirement. However, block-level mapping leads to huge garbage collection overhead. Many researches try to minimize the overhead by introducing hybrid mapping mechanism, such as FAST [18], LAST [17], and adaptive two-level management scheme proposed by Wu and Kuo[27]. Another approach to reduce SRAM requirement is to record address mapping by variant length [19]. When data with sequential logical address are stored successively, mapping table in SRAM only keeps the first location and the length of the following successive addresses. A new approach to utilize limited SRAM requirement and reserve performance of page-level mapping granularity is a demand-based selective caching mechanism [10] proposed by Gupta et al. They left the whole page-level mapping table in the Flash, and load needed partial mapping table to SRAM on demand. This approach successfully reduced garbage collection overhead and improved performance in enterprise-scale storage systems compare to hybrid mapping mechanisms. For garbage collection policies, Wu et al. explored the performance of greedy, locality gathering, and hybrid garbage collection policies under different circumstances [28]. Rosenblum et al. proposed cost-benefit policy [15], which is a value-based heuristic, to minimize the number and overhead of garbage collection. Garbage collection policy design can further considering wear-leveling effect. Chiang et al. design a Cost Age Times method [6], which selects the victim block according to cleaning cost, ages of live data, and the number of times the block has been erased. Kim et al. balanced garbage collection performance and blocks wearing by introduced a concept of leveling degree [14]. Larger leveling degree drives garbage collection to consider wear-leveling more, while less leveling degree focusing on minimize garbage collection overhead when choosing victim block. The leveling degree can be adjusted by circumstance. In addition to garbage collection policy, many wear-leveling researches adopt the concept of hot-cold swapping, first proposed by Chang and Kuo in 2005 [5], which moves cold data to the block that has been erased many times. STMicroeleconics [25] implements a two-level wear-leveling techniques. In the first level, new data are programmed to the free blocks that have the fewest erase count. When the difference between the maximum and the minimum number of erase count per block reaches a specific threshold, the second level wear-leveling is triggered, and the cold data is copied to the block with maximum erase count.

Recently, as SSD market growing, the concentration of researchers moves to the system-level design. Agrawal et al. [1] explored the design consideration of SSD, and discussed the internal organization. The authors first described the basic functionality that SSD must have, and then indicated the major challenges in implementation. They also evaluated some of the design choices, such as page size, over-provisioning, ganging, and striping. A larger page size increases number of read-modify-write, thus reduces the IO performance. Over-provisioning reduces Flash capacity, but improves garbage collection efficiency. Ganging offers the possibility of scaling capacity without linearly scaling pin density and firmware logic complexity. With regard to striping, they stripe requests to multiple packages by splitting up each request into parallel 4KB requests, but the data allocation policy is not mentioned. Shin et al. [24] target on FTL design exploration for server applications, and covered a variety of page mapping mechanisms. They evaluated static allocation by narrow striping and wide striping mapping mechanism, and found that wide striping mapping mechanism can achieve high parallelism and even distribution of requests in dominant sequential IO workload. They tested dynamic allocation with chip allocation pool and SSD allocation pool, and came up with the conclusion that if the workload is random dominant, larger allocation pool leads to higher parallelism and even distribution. They also investigated the effect of wear-leveling within different wear-leveling cluster size. Larger wear-leveling cluster size can even wear level throughout larger number of blocks, but come with larger overhead and response time. They concluded that for different workload characteristics, system requirements, and the remaining lifetime of SSDs, the decisions for designing customized FTLs can be varied. Dirik et al. [9] studied the interplay between SSD organization and performance. They explored in detail the system-level organization choices for SSDs, such as varying number of busses, speeds and widths of busses, and degree of concurrent access allowed on each bus, in the context of user-driven workloads. They found that NAND Flash memory performance is not limited by its serial interface, but its core interface – the movement of data between the Flash device's internal storage array and internal data register. Therefore, performance improvement by exploiting concurrency (e.g., using multiple independent banks combined with multiple independent channels) would be more significant than increasing bandwidth of serial interface.

In this thesis, I focus on different data allocation policy approaches and its performance impact on different Flash organization. The tradeoffs between striping and non-striping are discussed, and different striping unit size are evaluated to detail examined the tradeoffs.

Chapter 6

Conclusion

In this thesis, tradeoffs between striping and non-striping are discussed. With a lighter workload, the striping mechanism leads to a significant performance improvement proportional to the number of available banks. However, when the workload becomes heavier, non-striping has comparable performance to striping. According to the experimental results, the 2 PC workloads with IOPS greater than 27 is relatively heavy if we have only 4 Flash banks. Under this circumstance, non-striping induces similar response time as striping. Non-striping has another advantage: better garbage collection efficiency. Therefore, if garbage collection will be triggered under heavy workload, non-striping can have better performance than striping.

With detail examining in different striping unit size, I found that non-striping is not always the one with least erase count. If the trace contains various request length including some extremely large request, non-striping leads to imbalanced bank utilization. The ones with higher utilization trigger garbage collection frequently, and bring on large erase count.

Bibliography

[1] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy. Design tradeoffs for ssd performance. In *ATC'08: USENIX 2008 Annual Technical Conference on Annual Technical Conference*, pages 57–70, Berkeley, CA, USA, 2008. USENIX Association.

[2] A. M. Caulfield, L. M. Grupp, and S. Swanson. Gordon: using flash memory to build fast, power-efficient clusters for data-intensive applications. In *ASPLOS '09: Proceeding of the 14th international conference on Architectural support for programming languages and operating systems*, pages 217–228, New York, NY, USA, 2009. ACM.

[3] 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*, pages 1126–1130, New York, NY, USA, 2007. ACM.

[4] L.-P. Chang and T.-W. Kuo. An adaptive striping architecture for flash memory storage systems of embedded systems. In *RTAS '02: Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02)*, page 187, Washington, DC, USA, 2002. IEEE Computer Society.

[5] L.-P. Chang and T.-W. Kuo. Efficient management for large-scale flash-memory storage systems with resource conservation. *Trans. Storage*, 1(4):381–418, 2005.

[6] M.-L. Chiang, P. Lee, and R.-C. Chang. Managing flash memory in personal communication devices. In *Consumer Electronics, 1997. ISCE '97., Proceedings of 1997 IEEE International Symposium on*, pages 177–182, Dec 1997.

[7] S. Corporation. K9f8g08uxm flash memory specification. *Datasheet,http://www.samsung.com/global/system/business/semiconductor/product/2007/6/11/NANDFlash/SLC_LargeBlock/8Gbit/K9F8G08U0M/ds_k9f8g08x0m_rev10.pdf*, 2007.

[8] S. P. E. Corporation. Specweb2005. *http://www.spec.org/web2005/*, 2008.

[9] C. Dirik and B. Jacob. The performance of pc solid-state disks (ssds) as a function of bandwidth, concurrency, device architecture, and system organization. In *ISCA '09: Proceedings of the 36th annual international symposium on Computer architecture*, pages 279–289, New York, NY, USA, 2009. ACM.

[10] A. Gupta, Y. Kim, and B. Urgaonkar. Dftl: a flash translation layer employing demand-based selective caching of page-level address mappings. In *ASPLOS '09: Proceeding of the 14th international conference on Architectural support for programming languages and operating systems*, pages 229–240, New York, NY, USA, 2009. ACM.

[11] S.-g. L. Han-joon Kim. An effective flash memory manager for reliable flash memory space management. *IEICE Trans Inf Syst (Inst Electron Inf Commun Eng)*, E85-D(6):950–964, 2002.

[12] M.-S. Inc. Ide/sata/scsi ffd. *http://www.m-sys.com.*

[13] S. Inc. Silicondrive. *http://www.siliconsystems.com.*

[14] H.-J. Kim and S.-G. Lee. A new flash memory management for flash storage system. In *Computer Software and Applications Conference, 1999. COMPSAC '99. Proceedings. The Twenty-Third Annual International*, pages 284–289, 1999.

[15] J. Kim, J. M. Kim, S. Noh, S. L. Min, and Y. Cho. A space-efficient flash translation layer for compactflash systems. *Consumer Electronics, IEEE Transactions on*, 48(2):366–375, May 2002.

[16] N. Lee, Hyung Gyu; Chang. Low-energy heterogeneous non-volatile memory systems for mobile systems. *Journal of Low Power Electronics*, 1(1):52–62, 2005.

[17] S. Lee, D. Shin, Y.-J. Kim, and J. Kim. Last: locality-aware sector translation for nand flash memory-based storage systems. *SIGOPS Oper. Syst. Rev.*, 42(6):36–42, 2008.

[18] S.-W. Lee, D.-J. Park, T.-S. Chung, D.-H. Lee, S. Park, and H.-J. Song. A log buffer-based flash translation layer using fully-associative sector translation. *ACM Trans. Embed. Comput. Syst.*, 6(3):18, 2007.

[19] Y.-G. Lee, D. Jung, D. Kang, and J.-S. Kim. μ-ftl:: a memory-efficient flash translation layer supporting multiple mapping granularities. In

EMSOFT '08: Proceedings of the 8th ACM international conference on Embedded software, pages 21–30, New York, NY, USA, 2008. ACM.

[20] M-Systems. Two technologies compared: Nor vs. nand. *White Paper,http://www.dataio.com/pdf/NAND/MSystems/ MSystems_NOR_vs_NAND.pdf*, 2003.

[21] S. L. Min and E. H. Nam. Current trends in flash memory technology. In *Design Automation, 2006. Asia and South Pacific Conference on*, pages 2 pp.–, Jan. 2006.

[22] B. Networks. 'e-disk ata:2.5' ide/ata flash disk. *http://www.bitmicro.com.*

[23] C. Park, P. Talawar, D. Won, M. Jung, J. Im, S. Kim, and Y. Choi. A high performance controller for nand flash-based solid state disk (nssd). In *Non-Volatile Semiconductor Memory Workshop, 2006. IEEE NVSMW 2006. 21st*, pages 17–20, Feb. 2006.

[24] J.-Y. Shin, Z.-L. Xia, N.-Y. Xu, R. Gao, X.-F. Cai, S. Maeng, and F.-H. Hsu. Ftl design exploration in reconfigurable high-performance ssd for server applications. In *ICS '09: Proceedings of the 23rd international conference on Supercomputing*, pages 338–349, New York, NY, USA, 2009. ACM.

[25] STMicroelectronics. Wear leveling in single level cell nand flash memories. *Application Node (AN1822), http://www.eetindia.co.in/ARTICLES/2004NOV/A/2004NOV29 _MEM_AN09.PDF?SOURCES=DOWNLOAD*, 2006.

[26] D. Woodhouse. Jffs: The journalling flash file system. *Ottawa Linux Symposium*, 2001.

[27] C.-H. Wu and T.-W. Kuo. An adaptive two-level management for the flash translation layer in embedded systems. In *ICCAD '06: Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, pages 601–606, New York, NY, USA, 2006. ACM.

[28] M. Wu and W. Zwaenepoel. envy: a non-volatile, main memory storage system. In *ASPLOS-VI: Proceedings of the sixth international conference on Architectural support for programming languages and operating systems*, pages 86–97, New York, NY, USA, 1994. ACM.