國立交通大學

資訊學院　資訊學程

碩 士 論 文

兩層式寫入緩衝區的管理階層於固態硬碟上之應用

A Two-Level Write-Buffer Management Scheme for Solid-State Disks

研 究 生：林燕屏

指導教授：張立平　教授

中 華 民 國 九 十 八 年 三 月

兩層式寫入緩衝區的管理階層於固態硬碟上之應用
A Two-Level Write-Buffer Management Scheme for Solid-State Disks

研 究 生：林燕屏　　Student：Yen-Ping Lin

指導教授：張立平　　Advisor：Li-Pin Chang

國 立 交 通 大 學

資訊學院　資訊學程

碩 士 論 文

A Thesis

Submitted to College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Computer Science

Feburary 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年三月

兩層式寫入緩衝區的管理階層於固態硬碟上之應用

學生：林燕屏　　指導教授：張立平

國立交通大學　資訊學院　資訊學程碩士班

摘　要

快閃記憶體廣泛應用於嵌入式系統中，隨著快閃記憶體的容量快速提升以及成本下降，使用 NAND 快閃記憶體的固態硬碟(SSD)取代傳統硬碟，成為可攜式儲存裝備的最佳選擇。但由於 NAND 快閃記憶體的特性不同於一般快閃記憶體，現存的 FTL/NFTL 對於固態硬碟的管理並未有良好的效率。因此本篇論文設計了 Two-Level 的 Write Buffer 來管理寫入的資料，並將資料重整後寫入固態硬碟，不但能有效減少 NAND 快閃記憶體性能的劣化，更能有效地提升整體系統的效率。

A Two-Level Write-Buffer Management Scheme for Solid-State Disks

student：Yen-Ping Lin　　Advisors：Li-Pin Chang

Degree Program of Computer Science
National Chiao Tung University

ABSTRACT

NAND Flash Memory is gradually used in embedded system. As the increasing capacity and the descending cost of NAND flash memory, it will be a trend to use NAND flash memory in Solid State Disk (SSD) as portable storage disk instead of Hard Disk. We propose a Two-Level Write Buffer scheme to management the write data and write back to SSD after sort, this scheme will cover the deficiency of NAND flash memory and enhance the system performance.

誌　　謝

在交大的二年多來的學習即將劃下句點，感謝我的指導老師張立平教授，在嵌入式儲存領域上的專業指導，讓我獲得寶貴的經驗及建議。沒有張立平教授細心地指導，這篇論文無法順利地完成。

其次要感謝我的家人及親友們的關心；尤其是我的老公溥如，一直以來，從旁協助我及打理一切，讓我在家庭、事業及學業上無後顧之憂。在我沮喪時，不斷給我加油打氣，鼓舞著我，讓我能夠持續努力完成學業。

最後，僅將這份榮耀獻給我最愛的老公與家人，以及最敬愛的師長，感謝他們長久以來的支持與關心。

目　　錄

圖　目　錄

表　目　錄

一、Introduction

NAND 快閃記憶體是非揮發性記憶體，目前為嵌入式系統的主流儲存裝置，隨著 NAND 快閃記憶體密度的快速提升以及成本的下降，便有了使用 NAND 快閃記憶體的固態硬碟(SSD)取代傳統硬碟的方案。另外，由於 NAND 快閃記憶體式的固態硬碟並無傳統硬碟的機械磁頭，因此擁有比傳統硬碟更快速的讀寫速度、較低的耗電、輕薄好攜帶以及更好的可靠性。可預期地，固態硬體(SSD)在未來也將成為電腦產業中傳統硬碟的替代品。

因為 NAND 快閃記憶體是以 Page 為寫入單位，而以 Block 為抹除單位，另外有著 *erase-before-write*，即同個 page 不可以重覆寫入(overwrite)，也無法做 in-place update，必須要先寫到其他可用的空間來做 out-place update，因此 SSD 的存取方式和傳統硬碟不同，而需要一個 firmware 來管理 SSD。其中，NFTL 是 NAND 快閃記憶體上最廣泛使用的管理機制，包括 Address Translation、Garbage Collection 以及 Wear Leveling 三部分，將 NAND 快閃記憶體的物理特性隱藏起來，使得外部得以用傳統硬碟的方式對 NAND 快閃記憶體存取資料。

我們分析實際存取的狀況下，會有兩種寫入資料：小而隨機以及大而連續的寫入，當寫入資料時，會配置可用的空間來做 out-place update。資料為小而隨機的資料時，資料是一直重覆寫入同個紀錄區塊內，則會造成紀錄區塊滿了，而時常需做回收，但回收後隨即又寫滿了，一直在做無效的回收抹除。再者，配置的單位是以 block 為單位，小而隨機的寫入資料仍然配置一個 block 單位，容易造成空間搶奪的問題，讓可用空間的利用率低，抹除成本增加。由此可知 NFTL 對於小而隨機資料的處理能力是很差的。

若寫入大而連續的資料，不但讓可用空間的利用率提高，再者當寫入的連續資料寫滿一個 block 時，這時候新的 block 資料只需要和舊的 block 資料做對調，抹除的成本低，因此 NFTL 比較喜歡這種大而連續的寫入資料。

而當 NFTL 中的 Garbage Collection 以及 Wear Leveling 起動時，會造成對 NAND 快閃記憶體操作會有時間 latency，以及需要額外的空間讓 GC & WL 來作用，此為管理的主要成本，也是造成系統效率低落的原因。

本篇論文採取揮發性記憶體做為寫入緩衝區，這個緩衝區位於 SSD 內部，並使用兩層式的架構來做為緩衝區的管理階層，主要是希望減少 NFTL 的管理成本，而利用兩方面來達成：其一為減少資料寫入量，可透過弱化時間侷限性的資料，而有效地吸收 Random Write 資料，減少寫回 NAND 快閃記憶體時造成的 overhead。另一則將寫回 NAND 快閃記憶體的動作轉換為對 NFTL 友善的方式，可針對將空間侷限性的資料給最佳化，才將資料寫回 NAND 快閃記憶體中。

實驗部分以程式模擬實作本文所提到的兩層式寫入緩衝區系統，與 Log Block NFTL 及 Block LRU 的模擬做比較。由於兩層式寫入緩衝區系統主要目的

是減少 NAND 快閃記憶體做垃圾回收時造成的 erase operation，因此實驗主要測量的目標為 erase、switch erase 以 merge switch 的次數。而寫入緩衝區的有效利用也是評斷本系統的效能之一，故 page-level 的 write-back 次數、page-level 的 hit ratio 以及 block-level 的 space utilization 也會列為量測的目標。實驗的 Workload 為一般 NFTS 使用者在電腦上一般使用兩個月的 trace，透過我們的架構，在緩衝區為 24MB 時，可以將 erase count 減少為 Log NFTL 的 33.5%。

本論文的結構如下：第二章簡述相關文獻及描述問題，第三章說明 Two-Level Write-Buffer Management Scheme 的主要架構及方法的論述，第四章使用模擬的方法來評估系統效能，並和 Log Block NFTL 與 Block LRU 做比較。

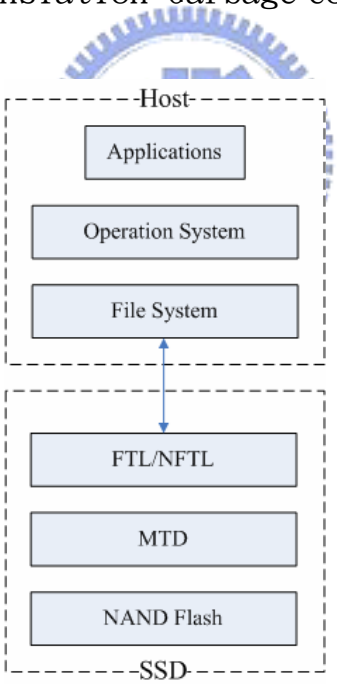
二、Motivation

由於 NAND 快閃記憶體已漸漸地使用於固態硬碟(SSD)上，此節會先介紹固態硬碟的特性，以及 SSD 裡對 NAND 快閃記憶體的管理機制，並描述相關的文獻。

2.1 SSD Architecture

固態硬碟內部有 controller、DMA 以及 NAND 快閃記憶體，其中 controller 主要是來控制實際上的儲存裝置 NAND 快閃記憶體，而有些 controller 為了存取 NAND 快閃記憶體時所用，會有揮發性記憶體。另外 DMA 則是負責和 Host 電腦傳收資料，增加傳收的資料流量。

固態硬碟的軟體上則由 NAND 快閃記憶體和快閃記憶體 Management 所組成。其中，Management 由兩個 Layer 所組成， *Memory Technology Device* (MTD) 以及 *Flash Translation Layer* (FTL)[1, 2]/*NAND Flash Translation Layer* (NFTL)[3]。MTD 為 Lower Software Layer，負責 NAND 快閃記憶體的讀、寫和抹除。FTL/NFTL 則是將 NAND 快閃記憶體模擬為 Block Device 及負責 NAND 快閃記憶體的管理，包括 Address Translation、Garbage Collection 以及 Wear Leveling 這三部分。



圖表 1 固態硬碟架構

NAND 快閃記憶體在操作上，讀寫以 page 為單位，抹除則以 block 為單位。同 1 個 page 不能做重覆寫入(overwrite)，每個 page 只能 write-once，其操作時間如表格 1 所示。另外，每個 block 都有它的耐用性(endurance)，因此每個 block 的寫入/抹除次數必須要平均，不然會很容易造成 block 的損壞。

表格 1 SLC[4]和 MLC[5]快閃記憶體操作時間

Operation	SLC	MLC
Page Size (byte)	2K+64	4K+128
Block Size (byte)	128K+4K	512K+16K
Block Erase (us)	1500	1500
Page Read (us)	25	60
Page Write (us)	200	800
Endurance (cycle)	100K	5K

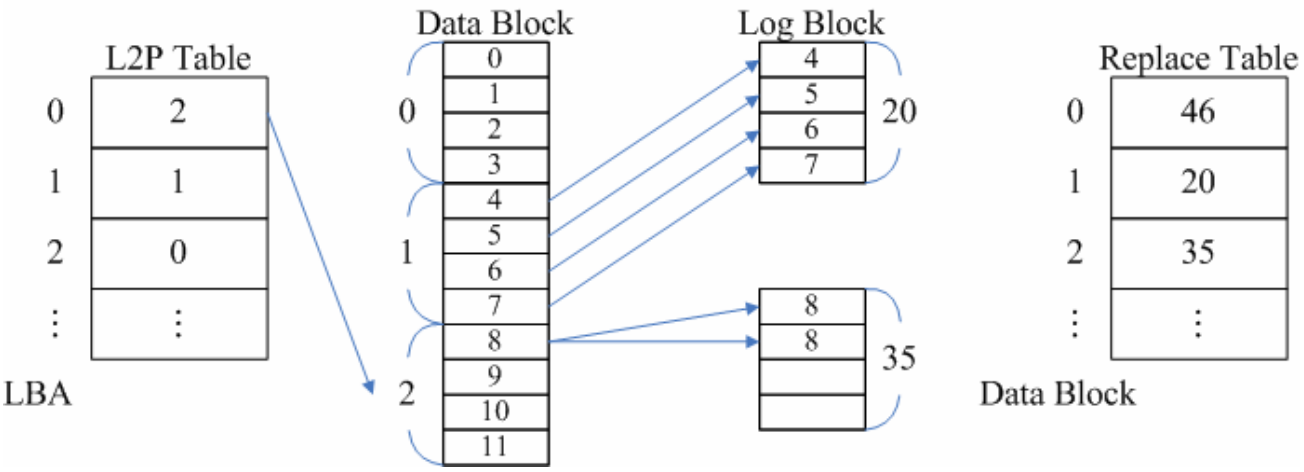
2.2 NFTL Management Issue

NAND 快閃記憶體其 page 寫入時間長(SLC:200us[4], MLC:800us[5])，以及 NFTL 在空間不夠時做 Garbage Collction，被回收的 blocks 抹除時間更長(SLC:1500us[4], MLC:1500us[5])，是造成 NAND 快閃記憶體管理上效率低落的最主要兩個原因，再加上 NAND 快閃記憶體上的每個 block 的抹除和寫入有使用次數的限制(SLC:100K[4], MLC:5K[5])，如何平均每個 block 的耐用性(Endurance)，將是 NFTL 管理上重要的議題之一。

2.2.1 Address Translation

NFTL 中的 Mapping Table 是採用改良的 Hybrid Mapping 的方式，結合 block Level 和 page level 的優點。採用 block level 的方式，加上有限數量的 log blocks 做為 page level 的 mapping。

我們將 NAND 快閃記憶體中 block level table 稱為 L2P table，它將 logical block address(LBA)對應到 NAND 快閃記憶體上的 physical block address，這些 physical blocks，我們將它命名為資料區塊(Data Block)。又因為 NAND 快閃記憶體無法直接 in-place 更新，因此我們找了一段可寫的 block，並將它取名為紀錄區塊(Log Block)。Block Level 的 Mapping Table，我們稱為 L2P Table，用來將 Logical Block Address(LBA)經過 L2P Table，轉換對應到資料區塊上的，當資料區塊裡已有資料了，便需要經由 Replace Table 找到此 LBA 對應的紀錄區塊來紀錄資料。



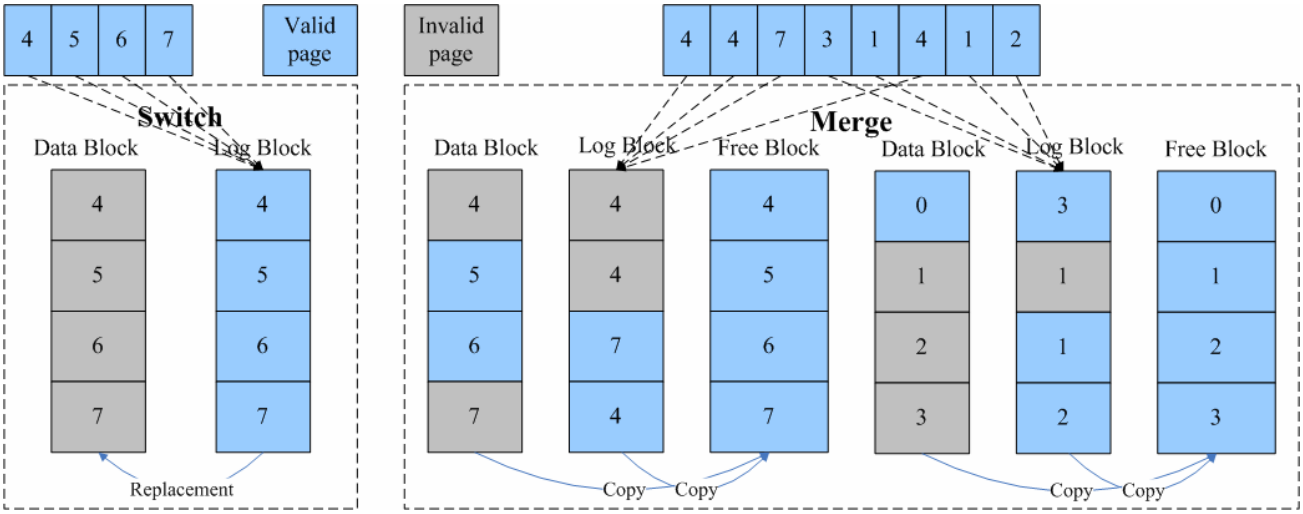
圖表 2 NFTL 的 Address Translation

2.2.2 Garbage Collection (GC)

將 NAND 快閃記憶體內不必要存在且浪費空間的 block 做垃圾回收，來增加可用的 block 數目。Block 做回收時，會根據資料寫入的順序，而影響到 NFTL 回收的效率。例如，寫入 sector 順序為 4->5->6->7 時，因為資料區塊資料已滿，便依序寫到紀錄區塊，做回收時，發現此紀錄區塊和資料區塊順序一樣，可以完全取代原有的資料區塊，因此只需將兩個 block 做 Switch，原來的紀錄區塊成為新的資料區塊，再抹除原本的資料區塊及更新 L2P、Replace 兩個 Table，便完成了垃圾回收。由此可知，做 Switch 的回收，只需要抹除一個 block 便行。

但若寫入 sector 的順序為 4->4->7->3->1->4->1->2 時，由於分別寫至屬於 Sector4~7 和 Sector0~3 的資料區塊已滿，因此各自寫入的紀錄區塊中。當要做回收時，若 Sector4~7 的資料區塊被選為 victim，發現它的紀錄區塊中並不連續，無法直接做 Switch，因此必須先找到一個 Free Block，做為 out-place 的區塊，並將資料區塊和紀錄區塊中，最新的 sector，分別 copy 到此 Free Block 中。此 Free Block 則為最新的資料，我們把此 Free Block 做為資料區塊，並回收資料區塊和紀錄區塊，並更新 L2P、Replace 的 Table，便完成了垃圾回收。可以知道，做 Merge 的回收，需要 copy 一個 block，抹除二個 block 才能完成回收。

可以知道，Switch 和 Merge 的回收，Merge 所花費的成本是高於 Switch 許多的。因此，垃圾回收除了能夠用最少的 cleaning work 來得到最多的 free blocks 之外，在回收時，如果能夠讓 Switch 回收的比例較高的話，對於整體系統效率的提昇也有實值的幫助。



圖表 3 Switch & Merge Operation

2.2.3 Wear Leveling

當使用 NAND 快閃記憶體時，會時常對某個檔案做修改，若每當此檔案更動時，便將資料寫回資料區塊或紀錄區塊，頻繁地更改此檔案，便會造成 NAND 快閃記憶體內的空間很快被使用完，而需要做垃圾回收。在做垃圾回收時，當有部分的區塊被重覆回收抹除後，這些區塊很容易因為過度寫入、抹除，而造成區塊的損毀。當 NAND 快閃記憶體使用的時間愈長，愈多的區塊損毀，以致於整個 NAND 快閃記憶體中可以使用的區塊愈來愈少，讓整個寫入的系統效率愈來愈低落。

為了防止這種情況，我們必須要使用 Wear Leveling，也就是平均每個區塊的磨損率，讓垃圾回收時，不會一直重覆挑選部分的區塊，造成損毀。但在做垃圾回收時，希望能用最少的 effort 來回收最多的 free block，因此在垃圾回收和平均磨損演算法之間是彼此衝突地，如何在垃圾回收和平均磨損演算法中取得平衡，是影響整體系統效率以及 NAND 快閃記憶體的耐用率很重要的關鍵。

2.3 Related Work

許多文獻在 NFTL 管理上的改進可以分成兩大方向，其一是在 FTL/NFTL[1-3]的架構上做改進，另一方面則是額外使用揮發性記憶體來當做 Write Buffer 使用，處理存取的資料，改善整體效能。

Log Block NFTL BAST 演算法[6]，將原本 NFTL 加上 Log Block 的架構，只是每個資料區塊只有一個紀錄區塊。因此，當寫入資料為 Random 的資料時，資料是一直重覆寫入同個紀錄區塊內，則會造成紀錄區塊滿了，而時常需做 Merge 的回收。由此可知 BAST 演算法對於 Random 資料的處理能力是很差的。

之後，便有針對 Random Write 資料處理的 Log Block NFTL FAST 演算法[7]，將 Log Block 分為兩種，一種是 SW Log Block，專門處理 Sequential Write 的資料，另一種則是 RW Log Block，則是針對 Random Write 的資料。每個資料區塊會有一個對應的 SW 紀錄區塊，但有 Sequential 的資料要寫入時，便從可用的 block list 中取得一個可用的區塊來拿，當 SW 紀錄區塊寫滿時，便可直接和資料區塊做 Switch。另外有部分的 RW 紀錄區塊是來紀錄任何 Random 寫入的資料，當要寫回時，會先選擇最舊的 RW 紀錄區塊為 victim，並掃描之後新的 RW 紀錄區塊，若有重覆的資料，便直接刪除此 victim 的資料。由此可知，RW 紀錄區塊雖然會減少和資料區塊 Merge 的回收，但是要掃描全部的 RW 紀錄區塊，是很花費時間的。

可以看出，與其在 NFTL 架構上的改變，不如直接增加一個揮發性記憶體來做為 Write Buffer，是比較的方法。*Flash aware buffer policy* (FAB)[8]則是寫入資料時以 page 為單位，從 free page list 中得到空的 page 來寫入，而 pages 中屬於同一個 block 的是屬於同一個 group 的，而這些 group 在空間不夠需要做回收時，則以 group 中擁有最多 page 數量的為 victim，來做垃圾回收。因為 FAB 主要是用在攜帶式的音樂裝置，因此它使用的 workload 會以 Sequential 的資料為主，因此這樣的演算法對於 Sequential 的資料有很好的處理方式，但要用在 SSD 上，有許多 Random Write 資料，就非常不適合了。

Block Padding Least Recently Used (BPLRU)[9]則是針對 Random Write 資料來處理，其做法是將揮發性記憶體以 block level 來管理，當某個 block 寫滿時，因為 Sequential 資料的機率是很高的，因此先 flush 回 NAND 快閃記憶體中。而當空間不夠時，以 LRU 為 order 來挑選 victim 的區塊。另外，為了避免 Merge 的回收，因此 flush 回 NAND 快閃記憶體時，皆是採用 page padding 的方式，也就是補滿 block 中未寫入的資料，以整個 block 寫回去，讓回收時可以都以 Switch 來做回收。可以發現的是，因為資料是 Random 和 Sequential 的資料都會寫進此 Write Buffer 中，Write Buffer 的空間大部分會被 Sequential 的資料給佔住，因此可以有效處理 Random 資料的空間其實並不大。

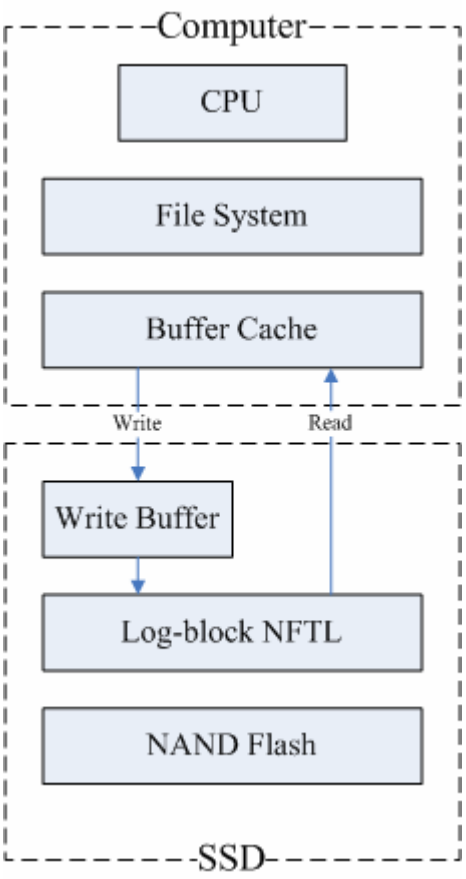
2.4 Idea

我們發現，與其去更改 NFTL 的架構，不如在 SSD 中增加揮發性記憶體來當 Write Buffer 是更好的方法。另外，因為現今 Host 電腦上，都已經有 Cache Buffer 了，因此與其使用 SSD 中有限大小的揮發性記憶體來處理 Read 資料，大容量的 Cache Buffer 就足以能夠吸收及處理 Read 的資料。因此，我們使用 SSD 中全部的揮發性記憶來處理 Write 的資料，只當成 Write Buffer 來使用，而 Read 資料會直接進入 FTL/NFTL[1-3]中。

我們的 Write Buffer 會在 NFTL 的上層，當 Host 電腦對 SSD 做寫入時，將由 SSD 內部的 controller 將資料寫至 Write Buffer 中，經由 Write Buffer

處理整合之後，才會將資料經由 NFTL 寫入實際的 NAND 快閃記憶體中。倘若 Host 電腦要讀 SSD 內部的資料時，則由 controller 直接經由 NFTL 對實際的 NAND 快閃記憶體做讀取。

另外，因為額外使用揮發性記憶體來當做 Write Buffer，buffer 的選用也是重要的考量之一。若選用的 buffer 尺寸過大的話，一方面會讓成本增加過多，另一方面會讓整體的耗電量過大，再者，尺寸過大時，要處理 buffer 內部資料結構而建構的 table、演算方法也會愈複雜。但 buffer 尺寸過小的話，無法有效地吸收 Random Write 資料，而讓系統效率根本無法提升。



圖表 4 Write Buffer 角色

三、Design and Implementation

3.1 Problem Definition

我們將使用揮發性記憶體來當成 Write Buffer，主要是希望減少 NAND 快閃記憶體做垃圾回收時造成的 erase 次數。而 Write Buffer 針對 Write Data 來處理，將會面臨的問題可以分成兩類。

一個是希望能夠減弱化資料的時間侷限性，有效地吸收 Random Write 資料，減少它寫回 NAND 快閃記憶體時造成的 overhead。如 NAND 快閃記憶體要做 GC 時，若資料都為 Random Write，則一定是以 Merge 的方法來回收，另外 NAND 快閃記憶體中的 Log Block 使用率也不會因為都被 Random Write 占用，而得被重覆回收造成 Block 的磨損。

另一個則是希望能強化資料的空間侷限性，將 Sequential Write 的資料，讓它更加地 Sequential，營造利於做 Switch 回收的條件，才將資料寫回 NAND 快閃記憶體中。

3.2 Technical Challenge

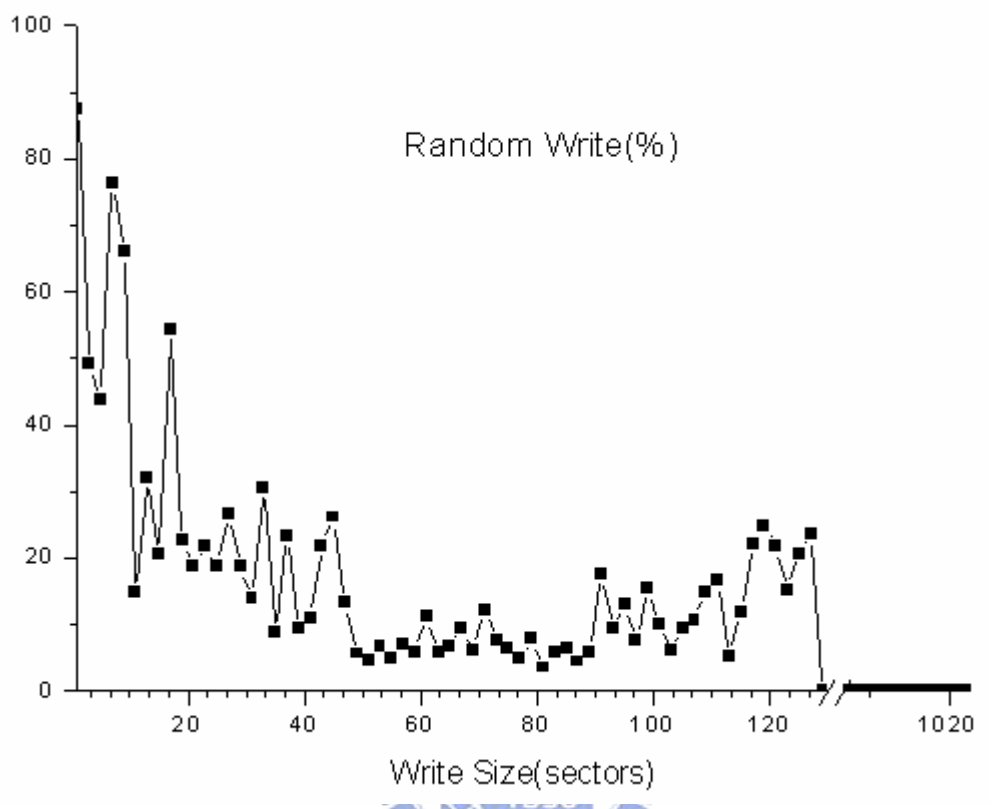
我們會將 Two-Level Write Buffer Scheme 遇到的挑戰，分成四方面來說明。第一節是先區分 Random Write 資料，第二節則是如何將寫入的資料有效地分開處理，第三節則是處理 Buffer Replacement 的準則，最後則是如何將 Write Buffer 的資料寫回 NAND 快閃記憶體中。

3.2.1 Identification of Random Writes

Random Write 資料為寫入資料不往前連續寫的資料則稱之為 random，此外不連續的資料會展現其時間區域性。Random 資料在 NFTL[1-3]的 block-level mapping 中，因為每次配置新空間仍然配置一個 block 單位，容易造成空間搶奪的問題，讓可用空間的利用率低，抹除成本增加，所以要特別做處理。而如何能有夠效地針對 Random Write 的資料來處理，首先就是必須先分辨出何者為 Random Write，以往的方法可以用 hash table[10]或者 LRU[11]的方法來辨認。LRU[11]使用兩個以 Block 為單位的固定長度 LRU list，當有 write 資料時，會去更動 LRU list 裡，利用資料更新的頻繁度而得知是否為 random 資料。而 hash table[10]則是讓每個 hash table 的 entry 有各自 Counter 值，當 write 資料寫入時，經過 multiple hash function 後，會將 hashed 的 entry Counter 值加一，而由此 Counter 值可以得知此 Block 是否為 random 資料。

不過我們觀察到實際上將 NTFS 使用者的 Workload 做分析後，並以 Write Size

來算出各種不同 Write Size 下，它的 Random Write 的資料占總資料的比例，並以%來表示。由圖表 5可以看出，在 Write Size 為 8 時，有 80%的資料為 Random Write，但在 Write Size 愈大時，Random Write 的比例不會超過 20%。我們可以很確切地知道，以 Write Size 來分辨 Random Write 是可行的。



圖表 5 Write 資料中 Random Write 的比例分布圖

3.2.2 Traffic Pattern Isolation

我們將 Write Buffer 分成 Block-level Buffer 和 Page-level Buffer 兩種，而分別以 block 和 page 為管理的單位。並以兩個 Buffer 來處理不同侷限性的資料，根據上一節將 Write 資料以 Size 來分成空間侷限性(spatial locality)以時間侷限性(temporal locality)的資料。用 Block-level Buffer 來處理空間侷限性的資料，讓資料更加 Sequential 後，才寫回 NAND 快閃記憶體中，若直接將 sequential 資料不做處理而直接寫入 NAND 快閃記憶體中，當寫入的資料未寫滿一個 Block 時 可以發現需要 padding 的成本，另外若 NAND 快閃記憶體中的 log block 都為未寫滿的資料，嚴重的時候會造成 log block threshing。而用 Page-level Buffer 來處理時間侷限性的資料，將 Random 資料弱化後，才寫回 NAND 快閃記憶體中。

另外，因為 Block-level Buffer 中是以處理空間侷限性為目的資料，但仍然會有約 25%的資料是 Random Write 的資料，因此在 Block-level Buffer 中仍然會有 Sequential 的資料被 Random 的資料推擠而提早寫回 NAND 快閃記憶體

中。而在 Page-level Buffer 中雖以處理時間侷限性的資料為目的，但仍會有約 20%的資料是 Sequential 的資料，而佔滿 Page-level Buffer 的空間，進而推擠了 Random 資料的空間。

再者，因為資料寫入 Page-level 或 Block-level 的 Buffer 時，為了要維持兩邊 buffer 內資料的一致性，避免有舊的資料覆蓋到新資料的問題，在資料要寫入到 Page-level 之前，會先去查看 Block-level 的 table 中，是否有屬於同一個 LBA 的資料，若有的話，則會將資料寫入 Block-level 的 buffer 中，就不會根據先前提到的準則來寫入資料。如此的做法，若兩邊的 buffer 中都存在同一筆資料時，可以知道，在 Block-level Buffer 中的資料必定為新的資料。

3.2.3 Buffer-replacement Policy

兩個 Buffer 的 replacement 機制為了處理不同侷限性的資料，因此在 replacement 上也有不同的方式。

Block-level Buffer 在 replacement 上有二個原則來處理，第一個是當 block 中寫滿資料時，此種 block 寫回 NAND 快閃記憶體時只要做 Switch 回收即可，因此選擇將這種 block 優先放到 LRU 的 tail，讓它能夠提前寫回 NAND 快閃記憶體中。我們會將連續寫入資料打中的 Block 放入 LRU 的 head，讓它能夠在 buffer 中繼續寫入，當此 Block 的資料已經不再連續了，則會因為 Block LRU 的順序慢慢排到 LRU tail。另外，當 Buffer 中的空間不夠時，需要做 replacement 的時候，則會以 Block LRU 的順序來寫回 NAND 快閃記憶體中，讓連續不再繼續發生的 block 會因為 Block LRU 的順序，被選為做 victim block。

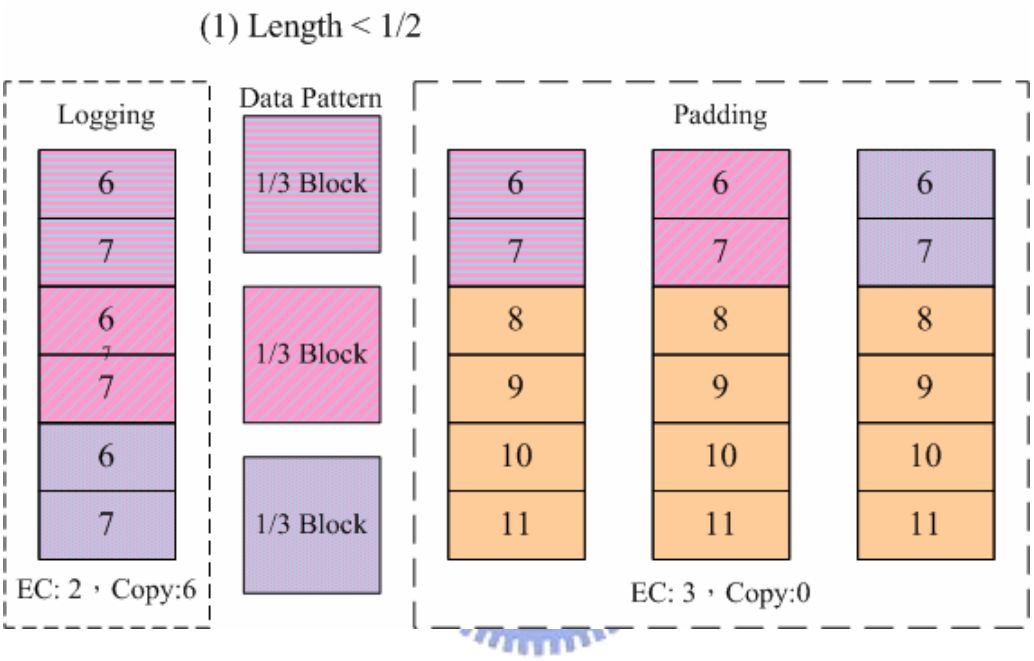
Page-level Buffer 是以 Page LRU 的順序來做 replacement，如此可以增加 random write 的資料停留在 Page-level Buffer 中的時間，當空間不夠需做 replacement 時，會選擇讓久未更新的資料來寫回 NAND 快閃記憶體中。將目前寫入的 Random 資料放至 LRU 的 head，讓它停留在 buffer 久些時間，來吸收相同的 Random 資料，而有效減少重覆資料寫入對系統效能的損耗。

3.2.4 Flash-write Policy

寫回 NAND 快閃記憶體時，根據 Block-level 和 Page-level，而寫回 NAND 快閃記憶體中的方法也有所不同。詳細如下

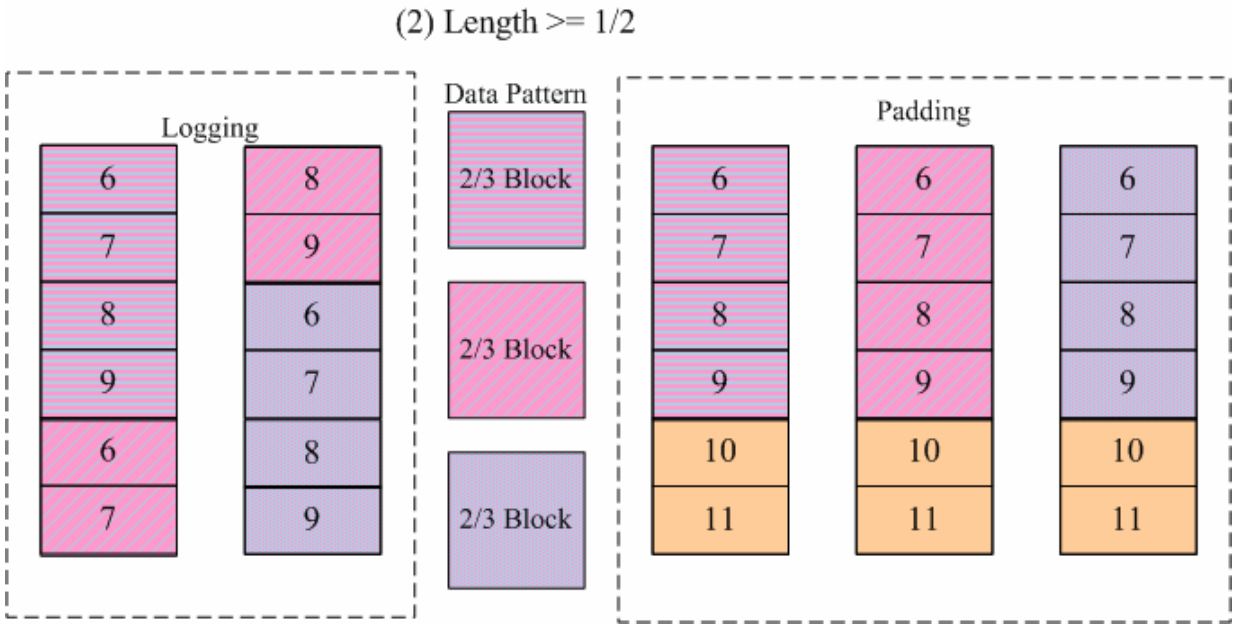
在 Block-level Buffer 中，當挑選到 victim 的 block 時，會根據 block 內寫的資料量而決定寫回的 policy。當 block 內的資料量少於 block 的一半時，例如圖表 6，有三筆資料要寫回，而每筆寫回的資料量只有 block 的 $\frac{1}{3}$ 時。當採

用 logging 的寫回，則每次寫滿了 block 的$\frac{1}{3}$，三筆資料後，會將此 block 寫滿，到了垃圾回收時，若此 block 被選為 victim 時，會採取 Merge 的回收，因此回收的 cost 為 erase 兩個 block(資料區塊和此紀錄區塊)，並且 copy 了六個 page 的資料。若採用 padding 的寫回方法，則每次都寫滿一個 block，做了 Switch 回收，則花費了 erase 一個 block，三筆資料的回收 cost 為 erase 三個 block。可以得知，當寫入資料量小於 block 的$\frac{1}{2}$時，使用 padding 的方式是比 較節省回收成本的。



圖表 6 寫回長度 < $\frac{1}{2}$ Block

反之，當寫入的資料量大於 block 的一半時，例如圖表 7所示，共寫回三筆資料，每筆資料量為 block 的$\frac{2}{3}$時。若採用 Logging 的方式時，當寫完第一筆資料時，block 還有$\frac{1}{3}$的空間，因此在寫第二筆資料時，會發生寫到一半就需要做回收的情況，這時候做了 Merge 回收，共 erase 二個 block，且 copy 六個 page 的花費。而後第二筆資料繼續寫入，當第三筆資料寫完時，若此紀錄區塊又被選為 victim 時，總共會花費 erase 四個 block，且 copy 十二個 page 的回收成本。反之，若直接採用 Padding 的方法，每寫完一筆資料，便花費 erase 一個 block 的 Switch 回收成本，三筆資料總共只會有 erase 三個 block 的回收成本，可以得知，在寫入資料量大於 block 的$\frac{2}{3}$時，直接採用 Padding 的方式是比較省成本的做法。



圖表 7 寫回長度 $\geq \frac{1}{2}$ Block

由此可知，Block-level Buffer 在針對寫回資料量的多寡，而我們會選擇最適宜的寫回方式，減少之後垃圾回收的成本。

而在 Page-level Buffer 中，因為 Page-level 本來就是要弱化時間侷限性的資料，因此被挑選的 victim Page，裡面的資料即是時間侷限性較低的資料了，因此會直接以 Logging 的方式寫回 NAND 快閃記憶體中。

而在 Page-level Buffer 中屬於同個 LBA 的 pages，未被選為 victim 的必定是屬於時間侷限性較強的資料，本來應該讓它繼續維持在 buffer 中。但因為寫回 NAND 快閃記憶體中的此 page，只占了紀錄區塊的一個 page，若此紀錄區塊被選為 victim 做垃圾回收的話，成本會相當地大，只使用了一個紀錄區塊來紀錄一個 page 的資料，相當浪費空間。因此我們會將屬於同個 LBA 的 pages 一併寫回 NAND 快閃記憶體中。

另外，因為考慮到 Page-level 和 Block-level 的同步性，因此 victim page 在寫回時，會去檢查 Block-level 中的 table，若 Block-level 中有資料，即表示此 victim page 的資料為舊的，所以我們會直接刪除此 page 內的舊資料。只有當 Block-level 中沒有此 page 的資料，才表示此 page 為需要更新回 NAND 快閃記憶體的資料。

3.2.5 Putting things together

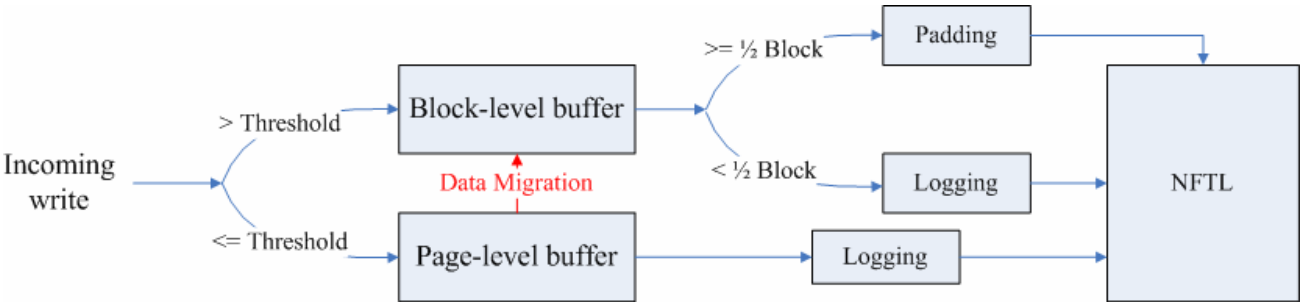
因此，我們整個 Two-Level Write-Buffer Management 的 Scheme 會如圖表 8所示，當有 write 資料進來我們的 Two-Level Write-Buffer 中時，為了維

持兩邊 buffer 內資料的一致性，避免有舊的資料覆蓋到新資料的問題，會先去檢查 Block-level 中是否有同個 LBA 的資料，若有的話，會將資料直接寫入 Block-level buffer 中，否則會以資料的大小來分出空間侷限性和時間侷限性的資料，而分別寫入 Block-level 和 Page-level 的 Buffer 中。如此的做法，若兩邊的 buffer 中都存在同一筆資料時，可以知道，在 Block-level Buffer 中的資料必定為新的資料。另外將資料移到 Block-level bufer 的原因是，當該 block 被寫回 NAND 快閃記憶體時將資料順便帶走。

而在 Block-level 和 Page-level 分別用 Block LRU 和 Page LRU 的 order 來做 replacement。為了兩個 buffer 的同步性，在寫回 NAND 快閃記憶體時，也會先互相去彼此的 table 中查看是否有相同的資料存在，若有的話，因為 Block-level 裡的 buffer 一定為新的，因此會將 Page-level 內舊的資料刪除才做寫回的動作。

另外，Page-level buffer 在寫回時，一律以 page 為單位，做 logging 的寫回。而 Block-level buffer 則依照 block 內的資料量而決定用 padding 或 logging 的寫回。

Write Buffer 的容量切分為 page-level 和 block-level buffer 的比例也會影響到整體的效率。當 block-level buffer 比例過多，而 page-level buffer 容量較小時，此時 page-level 無法有效吸收 Random Write 資料，仍然有一部分的資料會因 page-level buffer 的容量不夠而被擠回 NAND 快閃記憶體中，而造成整體效率提升效果不佳。但若 page-level 比例過多，而 block-level buffer 容量較少時，將使得 Sequential Write 資料無法有效地在 block-level buffer 中排序，而 page-level 容量在一定值以上，已經能夠完全吸收 Random Write 資料，因此過多的 page-level 容量只是浪費了使用空間。



圖表 8 Two-Level Write-Buffer Management Scheme

四、Experimental Result

4.1 Experimental Environment

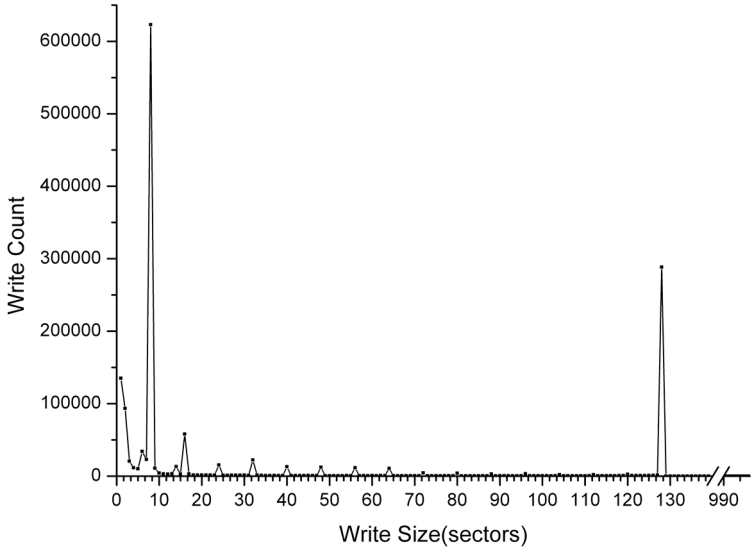
實驗部分我們是在一台筆記型電腦上模擬的數據,而模擬的 NAND 快閃記憶體是根據 MLC[5]的規格，將 Flash 設定為 21GB，每個 Block 大小為 512KB、Page 大小為 4KB、Sector 大小則為 512Byte。

表格 2 NAND 快閃記憶體的模擬系數

Flash Size	21GB
Block Size	512KB
Page Size	4KB
Sector Size	512 Byte

我們的 Workload 是在一台 Mobile PC 上，使用者存取 20GB 硬碟上收集而來的資料。Mobile PC 為 Windows XP，硬碟的 file system 為 NFTS 下收集一個月得到的 Workload，使用者在此 PC 做一般使用者在電腦上做的動作：上網、收發 email、播放電影、下載檔案、文書處理和遊戲。

我們分析此 Workload，可以發現，寫入的 Size 在 1~8 sectors 時數量是最多的，約占全部資料的 63%。另外，由圖表 5可以得知，在 1~8sectors 時，Random Write 的比例占了 68%，因此，我們認為將資料的 Threshold 定為 8 個 sectors 為合理的分割。而實驗的對象為使用傳統 BAST NFTL 而不使用 Write Buffer、使用本論文的兩層式 Write Buffer 加上 BAST NFTL，以及 BAST NFTL 加上只有 Block-level Write Buffer，此三種方式。



圖表 9 Workload 寫入 Size 分布圖

4.2 Performance Metrics

我們在模擬的測量上，會有六項數據來評斷。Erase Count、Switch Erase Count、Merge Erase Count、Page Buffer 寫回的數量、Page Buffer 的 Hit Ratio 以 Block Buffer 的使用率，並詳細如下。

Erase Count 為 NAND 快閃記憶體在做垃圾回收時，所抹除的區塊數量。此為 NAND 快閃記憶體效率最重要的測量，當此數據愈小的話，表示能夠不用做太多的垃圾回收，便可以處理同樣多寫入資料的處理，因此也就是代表整體系統效率的提昇。

Switch Erase Count。雖然 Erase Count 是最重要的指標，但當兩個方法的 Erase Count 相同時，若 Switch Erase Count 的比例較多，代表垃圾回收時做，紀錄區塊內的資料為連續的，因此只需將資料區塊和紀錄區塊做對調，然後抹除資料區塊，而不需將資料區塊和紀錄區塊中較新的資料 copy 到另一個 free 區塊去。因此 Switch Erase Count 的百分比愈高，也是系統效率提昇的另一指標。

Merge Erase Count 是表示在做垃圾回收時，紀錄區塊內的資料不是連續的，因此必須將資料區塊和紀錄區塊內最新的資料分別 copy 到一個 free 的區塊內，因此比 Switch 會花費更多的時間。因此，Merge Erase Count 和 Switch Erase Count 相反，當 Merge Erase Count 的百分比愈低，代表系統效率的提昇。

Page-level buffer 寫回 NAND 快閃記憶體的 page 數量，若寫回的數量愈少，代表 Page-level 良好地弱化了時間侷限性的資料，而減少了寫回 NAND 快閃

記憶體的數量，也讓 NAND 快閃記憶體能夠儲存更多的空間侷限性的資料。因此，寫回的數量愈少則愈減輕了系統做垃圾回收 Merge 的成本。

Page-level buffer 的 Hit Ratio 是指資料寫入 page-level buffer 時，是否為更新原有的資料，若有則為 Hit;反之，若需要新的空間來寫這筆新資料時，則稱為 miss。此數值愈高，可以得知 buffer 中吸收時間侷限性的資料愈多，buffer 是否能夠有效率地在處理時間侷限性的資料，因此 hit ratio 愈高，也代表 page-level buffer 的吸收時間侷限性的資料愈好。

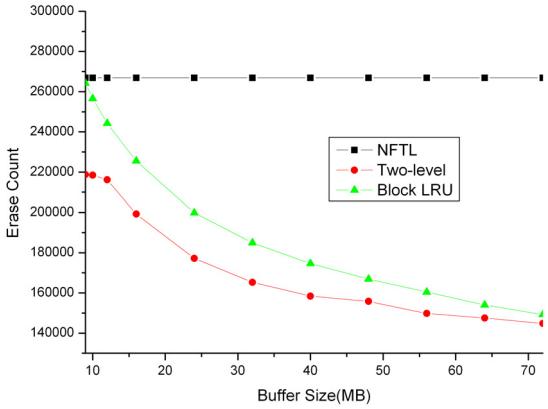
Block-level buffer 的空間使用率，是代表 block-level buffer 在寫回 NAND 快閃記憶體時，block 內的資料量是多少。若使用率愈高，代表 block-level buffer 愈強化了空間侷限性資料，才將這些資料一起寫回去。因此，我們會希望空間使用率愈高愈好。

4.3 Experimental Result

我們將 Block-level buffer 的容量設定為 8MB，而調整各種 Page-level buffer 的容量。我們將 NFTL 不加 Write Buffer 的實驗數據稱為 NFTL 的方式，Two-level 則為 NFTL 加上本論文兩層式的 Write Buffer，另外將 NFTL 加上只有 Block-level 的 Write Buffer 稱為 Block LRU 的方式。而實驗數據中的 X 軸為 Write Buffer 使用的容量，Y 軸則為量測的目標。

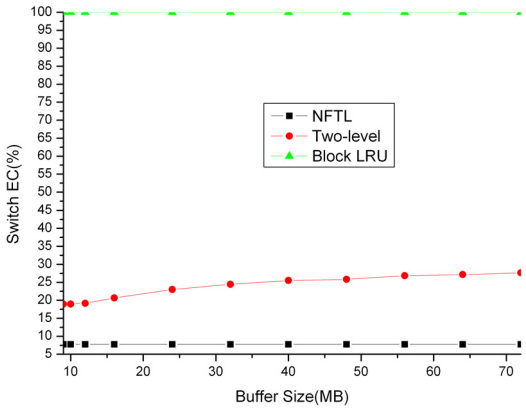
4.3.1 Erase Count、Switch Erase Count、Merge Erase Count

X 軸為 Write Buffer 容量，其中我們兩層式中的 Block Write Buffer 定為 8MB，而 Y 軸則 Erase Count 的 block 次數。由以下圖表 10可以看出，我們的 Two-Level 方法在 Erase Count 上比 NFTL 低了約 18~46%，整體系統效率改善的非常良好。另外，因為 Block LRU 的方法，將 Random 和 Sequential 資料混雜寫入 Block LRU 裡面，因此當容間不夠時，Random 資料會被 Sequential 資料擠壓出去，而頻繁地寫回 NAND 快閃記憶體中，再者每次寫回都是寫滿一整個 block，因此這種 Random 資料寫回造成寫滿一個 block 的情況，讓 Block LRU 的 Erase Count 比我們 Two-level 的方法多了約 3~21%。



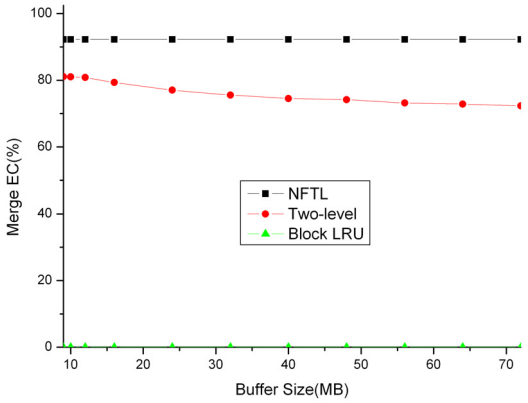
圖表 10 Erase Count

而在 Switch Erase Count 中，X 軸為 Write Buffer 容量，其中我們兩層式中的 Block Write Buffer 定為 8MB，而 Y 軸則為 Switch Erase Count 占 EC 的百分比。因為 Block LRU 都是採用寫回整個 block 的方式，因此全部都是 Switch 的回收方式。我們將 Two-Level 的方法和 NFTL 比較，發現能夠經由 Two-Level Buffer 的管理方式能將 Switch Erase Count 提高了約 11~21%，可以得知採用 Two-Level 可以有效地提昇 Switch 回收的比例。



圖表 11 Switch Erase Count

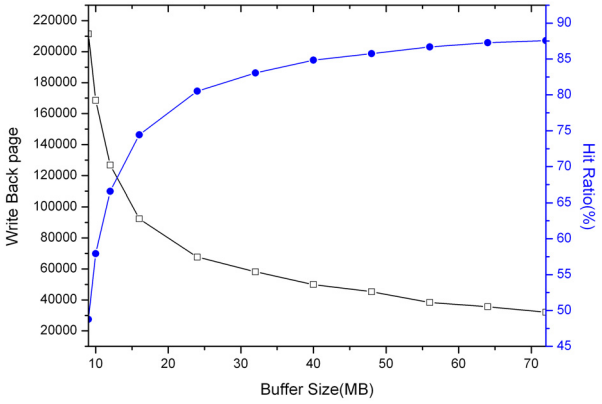
相反地，在 Merge Erase Count 的比例中，X 軸為 Write Buffer 容量，其中我們兩層式中的 Block Write Buffer 定為 8MB，而 Y 軸則為 Merge Erase Count 占 EC 的百分比。可以發現 Block LRU 都是整個 block 寫回的方式，因此 Merge 的比例為 0。而經由 Two-Level Buffer 的管理方式和 NFTL 比較時，能夠有效率地降低了 Merge Erase Count 比例約 11~21%。



圖表 12 Merge Erase Count

4.3.2 Page-level buffer ability

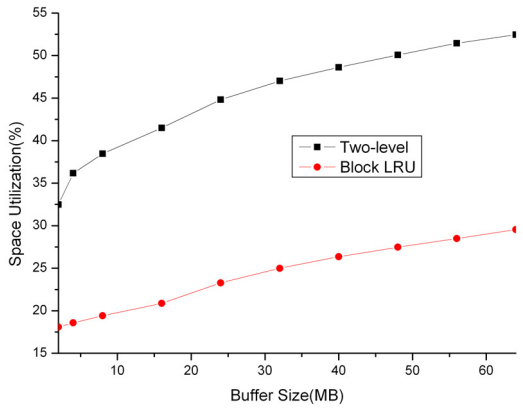
另外，我們會額外看 Two-Level Write-Buffer 中的 Page-level buffer，來看它的寫回 NAND 快閃記憶體的數量，可以得知它能否有效吸收時間侷限性的資料。X 軸為 Write Buffer 容量，其中我們兩層式中的 Block Write Buffer 定為 8MB，而 Y 軸則為 Page Buffer 寫回 NAND 快閃記憶體的 page 數量以及 Hit Ratio 的百分比。我們可以由圖表 13得知，隨著 page size 細微地增加，時間侷限性的資料被有效地吸收了。而 Page-level buffer 中的 Hit Ratio 在 page size 為 1MB 時，便有了 48%的 hit ratio，而之後急速上升至 87%左右。由這兩個數據可以知道我們 Page-level Buffer 中並沒有被過多的 Sequential 資料所佔據空間，而且有效率地吸收了時間侷限性的資料。



圖表 13 Page-level Buffer ability

4.3.3 Block-level Buffer ability

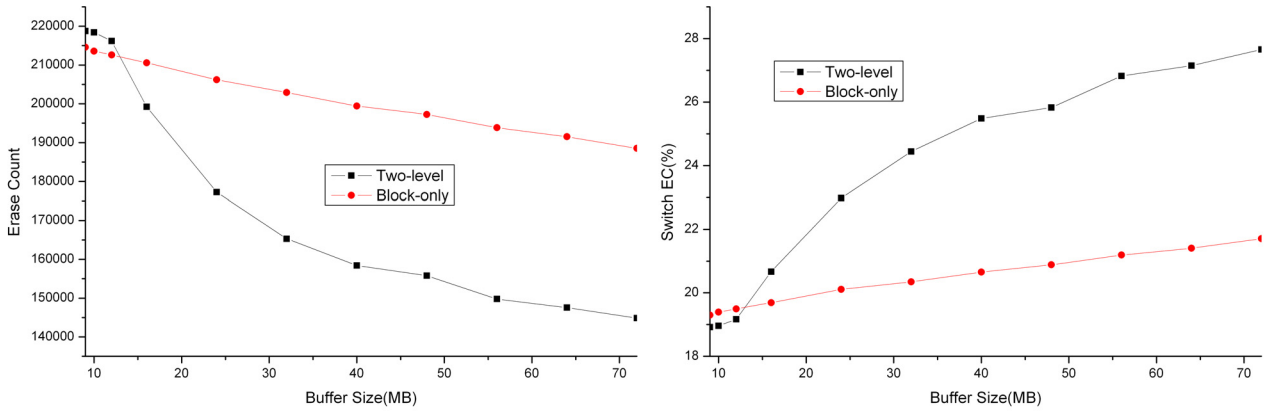
因為 NFTL 中並無使用 Block-level buffer，故我們和 Block LRU 比較，X 軸為 Write Buffer 容量，其中我們兩層式中的 Block Write Buffer 定為 8MB，而 Y 軸則為 Block Buffer 寫回 NAND 快閃記憶體時，資料量占此 victim block 的百分比。可以看出 Block LRU 的使用率相當地低，因為 block 的空間都被 Sequential 資料所佔滿，因此會排擠 Random 的資料，並將它寫回 NAND 快閃記憶體中，故使用率很低。但我們的 Two-Level 將空間侷限性高的資料寫回 Block-level buffer 中，故可看出我們的空間使用率比 Block LRU 提高了 14~23%。



圖表 14 Block Space Utilization

4.3.4 Compare with Block-only Buffer

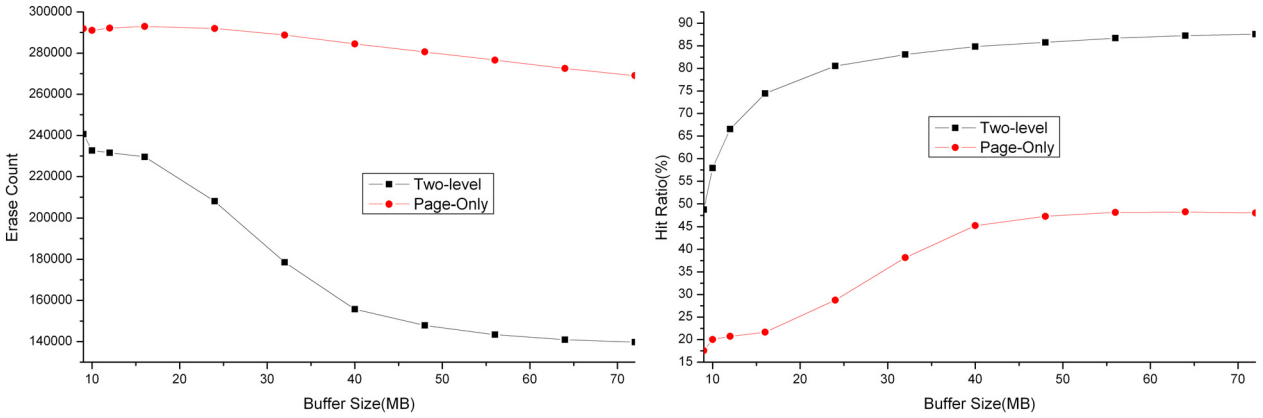
我們將 Two-Level 的方法，和全部資料一律寫進 Block-level buffer 來做比較，可以發現若資料全部寫進 Block-level buffer 裡，Sequential 的資料會一直被 Random 資料推擠出，因此可看出和 Two-Level 比較起來，Block-only buffer 的 Erase Count 多了 5~30%。Switch Erase Count 也無法隨著 buffer 容量加大而隨之提昇。



圖表 15 Compared Erase Count & Switch Erase Count

4.3.5 Compare with Page-only Buffer

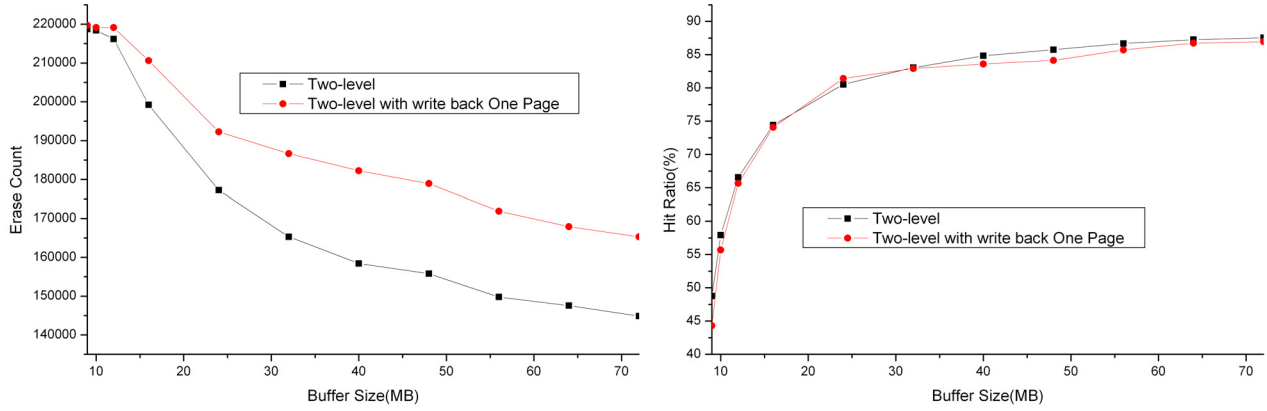
將 Two-Level 的方法，和所有的資料寫入 Page-level buffer 來做比較，可以知道所有資料進 Page-level buffer 時，buffer 的空間都被 Sequential 資料佔滿，因此一直推擠出 Random 資料，無法有效弱化時間侷限性的資料，所以 Erase Count 變得非常地差。相對的，Page 的 Hit Ratio 也很低。



圖表 16 Compared Erase Count & Page Hit Ratio

4.3.6 Page-level buffer Write Back Policy

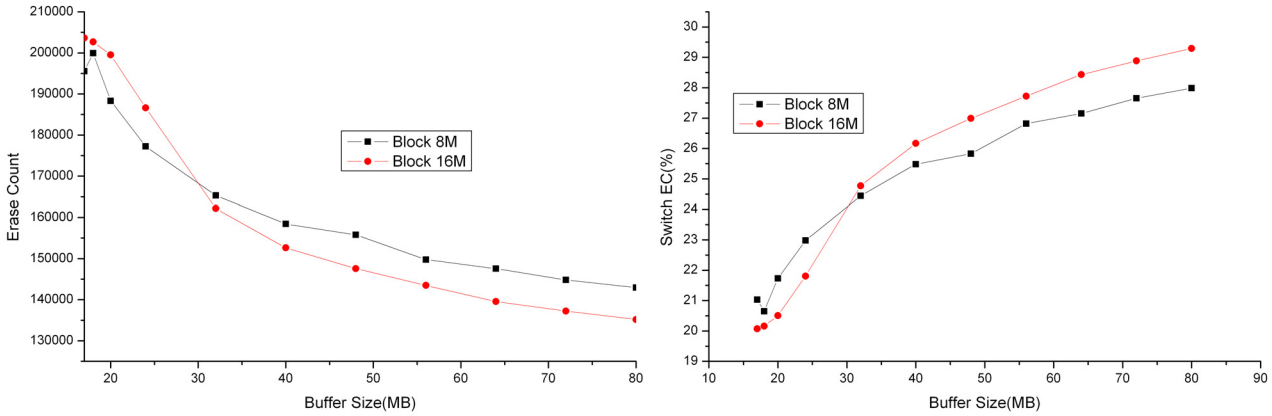
Two-Level 中的 Page-level buffer 在寫回 NAND 快閃記憶體時，會將屬於同樣 LBA 的 pages 一併寫回。若只寫回此 victim page 的話，可以由圖表 17 中看出來，Erase Count 會比 Two-Level 的方法多了 0~15%。雖然在 Page Hit Ratio 上，Two-Level 的方法會比只寫回 victim page 的方法多了不到 1%的差距，因此我們認為一併帶是有它的優勢在。



圖表 17 Compared Erase Count & Page Hit Ratio

4.3.7 Compare with different block-level buffer Size

我們若將 Write Buffer 中的 block-level buffer 容量變大為 16MB，page-level buffer 的容量會隨之縮減，可以由圖表 18中發現當 page-level buffer 的容量過小時，此時 page-level 無法有效吸收 Random Write 資料，仍然有一部分的資料會因 page-level buffer 的容量不夠而被擠回 NAND 快閃記憶體中，而 Erase Count 會變多了 5%左右，Switch Erase Count 的比例低了 1%左右。而 page-level 容量在一定值以上，已經能夠吸收大部分的 Random Write 資料，因此在 page-level buffer 大於 32MB 後，16MB 的 block-level buffer 的 Erase Count 和 Switch Erase Count 會比 8MB 的各改善了 5%及 1%。

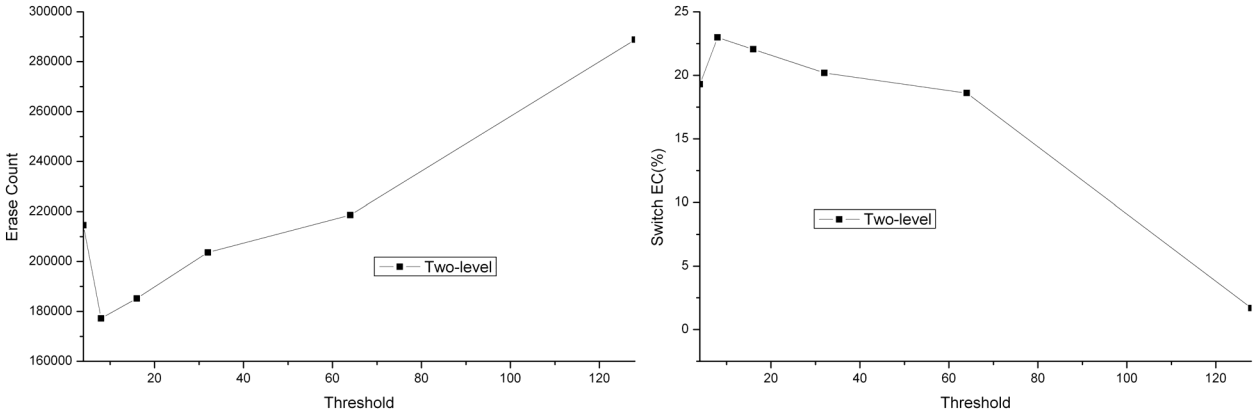


圖表 18 Compared Erase Count & Switch Erase Count

4.3.8 Compare with different Threshold

另外，X 軸使用不同的 threshold，Y 軸分別為 EC 以及 Switch EC 的百分比，其中 Block Write Buffer 定為 8MB，Page Write Buffer 定為 16MB。由圖表 19中可看到。Threshold 為 4 時，只有少量的 Write Data 進入 pge-level buffer 中，而大部分的 Random Data 都寫入 block-level buffer 中，因此 sequential 和 random 資料都在 block-level buffer 中，此時的 EC、Switch EC 很差。若 threshold 大於 8 時，較多的 Write Data 進入 page-level buffer 中，且資料中 Random Write 的資料比例變低了，因此 page-level buffer 在容量大於 4MB，即可以開始有效吸收 Random data 時，但因 Random Write 資料占 Write 資料中的比例較低，因此 page-level 並無法有效率地處理這些時間侷限性資料，而被空間侷限性的資料占滿了空間，導致 Erase Count 多了 4~10%，Switch Erase Count 的比例降低了 3~5%。而 Threshold 為 128 時，大部分的 sequential data 也都寫入 page-level buffer 中，因此 page-level buffer 中的 Random data 會被 Sequential data 所擠壓，page-level buffer 無法有效處理 Random data，

而讓 Erase Count 和 Switch Erase Count 變得非常地差。



圖表 19 Compared Erase Count & Switch Erase Count

五、Conclusion

我們可以發現,採用 Two-Level Write-Buffer Management Scheme 和 NFTL 以 Block LRU+NFTL 來比較，整體效率的提昇率非常地明顯，另外對於 NAND 快閃記憶體的磨損率和耐用性都有非常大的幫助。因此使用此 Two-Level Write-Buffer 的方式將兩種不同性質的資料分開，是很有效的方式。

但是，由於我們要維持 Block-level 和 Page-level 的同步性，因此有部分應該寫入 Page-level 的資料,卻為了解決同步性問題,而改寫入 Block-level buffer 裡，故在 Block-level 裡的資料，會有部分的 Random 資料和 Sequential 資料混雜的情況，也是造成 Block Utilization 沒辦法再提昇的瓶頸。

另外，我們使用揮發性記憶體當做 Write Buffer 使用，但在本實驗中，並未考慮到系統在突然斷電時，揮發性記憶體內的資料會全部消失，而沒有將更新的資料寫回 NAND 快閃記憶體中。因此可以使用非揮發性的記憶體如相變記憶體或鐵電記憶體當做 Write Buffer 來使用，解決此問題。

參 考 文 獻

[1] A. Ban, "Flash file system," M-Systems Flash Disk Pioneers Ltd., 1995.

[2] Intel, "Understanding the Flash Translation Layer (FTL) Specification," 1998.

[3] A. Ban, "Flash file system optimized for page-mode flash technologies," M-Systems Flash Disk Pioneers Ltd., 1999.

[4] "K9WAG08U1A 2G x 8 Bit NAND Flash Memory DataSheet," Samsung Electronics Company, 2006.

[5] "K9GAG08U0M 2G x 8 Bit NAND Flash Memory DataSheet," Samsung Electronics Company, 2006.

[6] K. Jesung, K. Jong Min, S. H. Noh, M. Sang Lyul, and C. Yookun, "A space-efficient flash translation layer for CompactFlash systems," *Consumer Electronics, IEEE Transactions on,* vol. 48, pp. 366-375, 2002.

[7] 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," *Trans. on Embedded Computing Sys.,* vol. 6, p. 18, 2007.

[8] J. Heeseung, K. Jeong-Uk, P. Seon-Yeong, K. Jin-Soo, and L. Joonwon, "FAB: flash-aware buffer management policy for portable media players," *Consumer Electronics, IEEE Transactions on,* vol. 52, pp. 485-493, 2006.

[9] H. Kim and S. Ahn, "BPLRU: a buffer management scheme for improving random writes in flash storage," in *Proceedings of the 6th USENIX Conference on File and Storage Technologies* San Jose, California: USENIX Association, 2008.

[10] J.-W. Hsieh, T.-W. Kuo, and L.-P. Chang, "Efficient identification of hot data for flash memory storage systems," *Trans. Storage,* vol. 2, pp. 22-40, 2006.

[11] C. Li-Pin and K. Tei-Wei, "An adaptive striping architecture for flash memory storage systems of embedded systems," in *Real-Time and Embedded Technology and Applications Symposium, 2002. Proceedings. Eighth IEEE*, 2002, pp. 187-196.